



NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

**FUZZ TESTING OF INDUSTRIAL NETWORK
PROTOCOLS IN PROGRAMMABLE LOGIC
CONTROLLERS**

by

James J. Gormley III

December 2017

Thesis Advisor:
Co-Advisor:

Thuy D. Nguyen
Cynthia Irvine

Approved for public release. Distribution is unlimited.

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			<i>Form Approved OMB No. 0704-0188</i>	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington, DC 20503.				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE December 2017		3. REPORT TYPE AND DATES COVERED Master's thesis
4. TITLE AND SUBTITLE FUZZ TESTING OF INDUSTRIAL NETWORK PROTOCOLS IN PROGRAMMABLE LOGIC CONTROLLERS			5. FUNDING NUMBERS	
6. AUTHOR(S) James J. Gormley III				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			10. SPONSORING / MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB number ____N/A____.				
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.			12b. DISTRIBUTION CODE	
13. ABSTRACT (maximum 200 words) Daily operations of U.S. Navy afloat and ashore systems are heavily reliant on industrial control systems (ICSs) to manage critical infrastructure services. Programmable logic controllers (PLCs) are vital components in these cyber-physical systems. The industrial network protocols used to communicate between nodes in a control network are complex and vulnerable to a myriad of cyber attacks, as reported by Department of Homeland Security Industrial Control Systems Cyber Emergency Response Team. This thesis utilizes protocol fuzz testing techniques to investigate potential vulnerabilities in the Allen-Bradley/Rockwell Automation (AB/RA) MicroLogix 1100 PLC through its implementation of EtherNet/IP, Common Industrial Protocol (CIP), and Programmable Controller Communication Commands (PCCC) communication protocols. This research also examines whether cross-generational vulnerabilities exist in the more advanced AB/RA ControlLogix 1756-L71 PLC. Our results discover several deviations from the EtherNet/IP and PCCC specifications in the MicroLogix 1100 implementation of these protocols. Additionally, we find that a recently disclosed denial-of-service vulnerability that renders the MicroLogix 1100 inoperable does not trigger a similar fault condition in the ControlLogix PLC.				
14. SUBJECT TERMS industrial control system, protocol fuzz testing, PLC, EtherNet/IP, CIP, PCCC, MicroLogix, ControlLogix			15. NUMBER OF PAGES 175	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UU	

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

**FUZZ TESTING OF INDUSTRIAL NETWORK PROTOCOLS IN
PROGRAMMABLE LOGIC CONTROLLERS**

James J. Gormley III
Lieutenant Commander, United States Navy
B.S., Villanova University, 2005
M.P.S., The George Washington University, 2013

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN CYBER SYSTEMS AND OPERATIONS

from the

**NAVAL POSTGRADUATE SCHOOL
December 2017**

Approved by: Thuy D. Nguyen
Thesis Advisor

Dr. Cynthia Irvine
Co-Advisor

Dr. Dan Boger
Chair, Department of Information Sciences

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Daily operations of U.S. Navy afloat and ashore systems are heavily reliant on industrial control systems (ICSs) to manage critical infrastructure services. Programmable logic controllers (PLCs) are vital components in these cyber-physical systems. The industrial network protocols used to communicate between nodes in a control network are complex and vulnerable to a myriad of cyber attacks, as reported by Department of Homeland Security Industrial Control Systems Cyber Emergency Response Team. This thesis utilizes protocol fuzz testing techniques to investigate potential vulnerabilities in the Allen-Bradley/Rockwell Automation (AB/RA) MicroLogix 1100 PLC through its implementation of EtherNet/IP, Common Industrial Protocol (CIP), and Programmable Controller Communication Commands (PCCC) communication protocols. This research also examines whether cross-generational vulnerabilities exist in the more advanced AB/RA ControlLogix 1756-L71 PLC. Our results discover several deviations from the EtherNet/IP and PCCC specifications in the MicroLogix 1100 implementation of these protocols. Additionally, we find that a recently disclosed denial-of-service vulnerability that renders the MicroLogix 1100 inoperable does not trigger a similar fault condition in the ControlLogix PLC.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I.	INTRODUCTION.....	1
A.	MOTIVATION	1
B.	RELEVANCE TO THE NAVY.....	2
C.	OBJECTIVES	3
D.	THESIS ORGANIZATION.....	3
II.	BACKGROUND	5
A.	COMMON INDUSTRIAL PROTOCOL (CIP)	5
B.	ETHERNET/INDUSTRIAL PROTOCOL (ETHERNET/IP)	7
C.	PROGRAMMABLE CONTROLLER COMMUNICATION COMMANDS (PCCC)	8
D.	FUZZ TESTING	9
E.	ICS FUZZERS	10
F.	SCAPY	10
G.	PREVIOUS SCAPY-BASED FUZZING.....	10
H.	ALLEN-BRADLEY / ROCKWELL AUTOMATION PLCS (MICROLOGIX 1100 AND CONGROLLOGIX 5570).....	11
III.	DESIGN	13
A.	OBJECTIVES	13
B.	METHODOLOGY	13
C.	TEST ENVIRONMENT	17
IV.	IMPLEMENTATION AND TEST PLAN.....	19
A.	FUZZER IMPLEMENTATION	19
1.	FUZZER MODIFICATIONS FOR MICROLOGIX.....	19
2.	FUZZER MODIFICATIONS FOR CONTROLLOGIX.....	20
B.	ENIP FUZZING TEST PLAN.....	20
1.	ENIP ListServices Command	22
2.	ENIP UnRegisterSession Command	23
3.	ENIP SendRRData Command.....	25
4.	ENIP SendUnitData Command.....	28
5.	ENIP Reserved for Legacy Use Commands	30
6.	ENIP Reserved for Future Use Commands.....	31
C.	CIP FUZZING TEST PLAN	33
1.	CIP Get_Attributes_All.....	34
2.	CIP Get_Attribute_List.....	35

3.	CIP Get_Attribute_Single	36
4.	CIP Find_Next_Object_Instance.....	38
D.	PCCC FUZZING TEST PLAN	40
1.	PCCC Echo Command.....	42
2.	PCCC Protected Typed File Read.....	44
3.	PCCC Protected Typed File Write.....	45
4.	PCCC Protected Typed Logical Write with Three Address Fields	47
5.	PCCC Unprotected Read	49
6.	PCCC Diagnostic Status.....	51
7.	PCCC Read Diagnostic Counters.....	52
8.	PCCC Restart.....	54
9.	PCCC Download Completed	56
10.	PCCC Protected Logical Read with Three Address Fields Command on ControlLogix	58
V.	TEST ANALYSIS	61
A.	ENIP TEST RESULTS	61
1.	ENIP ListServices Results.....	64
2.	ENIP UnRegisterSession Results.....	64
3.	ENIP SendRRData Results	66
4.	ENIP SendUnitData Results	68
5.	ENIP Reserved for Legacy Use Results	68
6.	ENIP Reserved for Future Use Results.....	69
B.	CIP TEST RESULTS	69
1.	CIP Get_Attributes_All Results	70
2.	CIP Get_Attribute_List Results	72
3.	CIP Get_Attribute_Single Results.....	74
4.	CIP Find_Next_Object_Instance Results	74
C.	PCCC TEST RESULTS	75
1.	PCCC Echo Results	78
2.	PCCC Protected Typed File Read Results	79
3.	PCCC Protected Typed File Write Results	79
4.	PCCC Protected Logical Write with Three Address Fields Results.....	79
5.	PCCC Unprotected Read Results.....	79
6.	PCCC Diagnostic Status Results	80
7.	PCCC Read Diagnostic Counters Results	80
8.	PCCC Restart Results	80
9.	PCCC Download Completed Results.....	81

10.	PCCC Protected Logical Read with Three Address Fields on ControlLogix Results.....	81
D.	DISCUSSION	82
VI.	CONCLUSION AND FURTHER WORK.....	85
A.	SUMMARY	85
B.	FUTURE WORK.....	86
APPENDIX A.	ENIP COMMAND RESPONSES.....	87
A.	ENIP LISTSERVICES TEST CASES.....	87
B.	ENIP UNREGISTERSESSION TEST CASES.....	91
C.	ENIP SENDRRDATA TEST CASES	94
D.	ENIP SENDUNITDATA TEST CASES.....	99
E.	ENIP RESERVED FOR LEGACY USE TEST CASES.....	104
F.	ENIP RESERVED FOR FUTURE USE TEST CASES	105
APPENDIX B.	CIP COMMAND RESPONSES.....	107
A.	CIP GET_ATTRIBUTES_ALL TEST CASES	107
B.	CIP GET_ATTRIBUTE_LIST TEST CASES	109
C.	CIP GET_ATTRIBUTE_SINGLE TEST CASES	116
D.	CIP FIND_NEXT_OBJECT_INSTANCE TEST CASES.....	120
APPENDIX C.	PCCC COMMAND RESPONSES	125
A.	PCCC ECHO TEST CASES	125
B.	PCCC PROTECTED TYPED FILE READ TEST CASES	130
C.	PCCC PROTECTED TYPED FILE WRITE TEST CASES.....	132
D.	PCCC PROTECTED LOGICAL WRITE WITH THREE ADDRESS FIELDS TEST CASES	134
E.	PCCC UNPROTECTED READ TEST CASES	137
F.	PCCC DIAGNOSTIC STATUS TEST CASES.....	138
G.	PCCC READ DIAGNOSTIC COUNTERS TEST CASES.....	139
H.	PCCC RESTART TEST CASES	141
I.	PCCC DOWNLOAD COMPLETED TEST CASES	141
J.	PCCC PROTECTED LOGICAL READ WITH THREE ADDRESS FIELDS TEST CASES	142
LIST OF REFERENCES	145
INITIAL DISTRIBUTION LIST	149

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

Figure 1.	CIP Network Architecture Stack. Source: [15].....	6
Figure 2.	CIP Object Model. Source: [23].	7
Figure 3.	EtherNet/IP Packet Encapsulation. Source: [25].	8
Figure 4.	MicroLogix 1100 PLC. Source: [39].	11
Figure 5.	ControlLogix PLC. Source: [40].....	12
Figure 6.	MicroLogix Testing Environment	18
Figure 7.	ControlLogix Testing Environment.....	18
Figure 8.	An Example ENIP ListServices Request over TCP Packet.....	22
Figure 9.	Hexadecimal View of Example ENIP ListServices Request over TCP Packet.....	23
Figure 10.	An Example ENIP ListServices Request over UDP Packet	23
Figure 11.	Hexadecimal View of Example ENIP ListServices Request over UDP Packet.....	23
Figure 12.	An Example ENIP UnRegisterSession Request over TCP Packet	24
Figure 13.	Hexadecimal View of Example TCP ENIP UnRegisterSession Request Packet	25
Figure 14.	An Example ENIP UnRegisterSession Request over UDP Packet.	25
Figure 15.	Hexadecimal View of Example ENIP UnRegisterSession Request over UDP Packet.....	25
Figure 16.	An Example ENIP SendRRData Request over TCP with an Encapsulated CIP Forward Open Request.....	27
Figure 17.	Hexadecimal View of Example ENIP SendRRData Request over TCP Packet.....	28
Figure 18.	An Example ENIP SendUnitData Request over TCP with an Encapsulated CIP Get_Attribute_All Request.....	29

Figure 19.	Hexadecimal View of Example EtherNet/IP SendUnitData Request over TCP Packet	29
Figure 20.	An Example ENIP Legacy Use Request over TCP	30
Figure 21.	Hexadecimal View of Example ENIP Legacy Use Request over TCP Packet.....	30
Figure 22.	An Example ENIP Legacy Use Request over UDP.....	31
Figure 23.	Hexadecimal View of Example ENIP Legacy Use Request over UDP Packet.....	31
Figure 24.	An Example ENIP Future Use Request over TCP.....	32
Figure 25.	Hexadecimal View of Example ENIP Future Use Request over TCP Packet.....	32
Figure 26.	An Example UDP ENIP Future Use Request	32
Figure 27.	Hexadecimal View of Example UDP ENIP Future Use Response Packet.....	32
Figure 28.	An Example CIP Get_Attributes_All Request over TCP	34
Figure 29.	Hexadecimal View of Example CIP Get_Attributes_All Request over TCP Packet	34
Figure 30.	An Example CIP Get_Attribute_List Request over TCP	35
Figure 31.	Hexadecimal View of Example CIP Get_Attribute_List Request over TCP Packet	36
Figure 32.	An Example CIP Get_Attribute_Single Request over TCP	37
Figure 33.	Hexadecimal View of Example CIP Get_Attribute_Single Request over TCP Packet	37
Figure 34.	An Example CIP Find_Next_Object_Instance Request over TCP.....	39
Figure 35.	Hexadecimal View of Example CIP Find_Next_Object_Instance Request over TCP Packet.....	39
Figure 36.	An Example PCCC Echo Request with Two Data Bytes over TCP.....	43
Figure 37.	Hexadecimal View of Example PCCC Echo Request over TCP Packet.....	43

Figure 38.	An Example PCCC Protected Typed File Read Request.....	44
Figure 39.	Hexadecimal View of Example PCCC Protected Typed File Read Packet.....	45
Figure 40.	An Example PCCC Protected Typed File Write Request.....	46
Figure 41.	Hexadecimal View of Example PCCC Protected Typed File Write Packet.....	46
Figure 42.	An Example PCCC Protected Typed Logical Write with Three Address Fields Request.....	48
Figure 43.	Hexadecimal View of Example PCCC Protected Typed Logical Write with Three Address Fields Request over TCP Packet	49
Figure 44.	An Example PCCC Unprotected Read Request	50
Figure 45.	Hexadecimal View of Example PCCC Unprotected Read Packet	50
Figure 46.	An Example PCCC Diagnostic Status over TCP Request.....	51
Figure 47.	Hexadecimal View of Example PCCC Diagnostic Status Request over TCP Packet	52
Figure 48.	An Example PCCC Read Diagnostic Counters Request	53
Figure 49.	Hexadecimal View of Example PCCC Read Diagnostic Counters Request Packet.....	54
Figure 50.	An Example PCCC Restart Request	55
Figure 51.	Hexadecimal View of Example PCCC Restart Request Packet	56
Figure 52.	An Example PCCC Download Completed Request	57
Figure 53.	Hexadecimal View of Example PCCC Download Completed Request Packet.....	57
Figure 54.	An Example PCCC Protected Logical Read with Three Address Fields Request.....	59
Figure 55.	Hexadecimal View of Example PCCC Protected Logical Read with Three Address Fields Request Packet.....	59
Figure 56.	ListServices Response over TCP (Fuzzed Session Handle).....	87
Figure 57.	ListServices Response over UDP (Fuzzed Session Handle)	88

Figure 58.	ListServices Response over TCP (Fuzzed Status)	88
Figure 59.	ListServices Response over UDP (Fuzzed Status)	89
Figure 60.	ListServices Response over TCP (Fuzzed Sender Context)	89
Figure 61.	ListServices Response over UDP (Fuzzed Sender Context)	90
Figure 62.	ListServices Response over TCP (Fuzzed Options)	90
Figure 63.	ListServices Response over UDP (Fuzzed Options)	91
Figure 64.	UnRegisterSession Response over TCP (Fuzzed Session Handle)	91
Figure 65.	UnRegisterSession Response over TCP (Fuzzed Status)	92
Figure 66.	UnRegisterSession Response over TCP (Fuzzed Sender Context)	92
Figure 67.	UnRegisterSession Response over TCP (Fuzzed Options)	93
Figure 68.	CIP Forward Open Response Following ENIP UnRegisterSession Request with Fuzzed Options Field	93
Figure 69.	UnRegisterSession Response over UDP (Functionality Test).....	94
Figure 70.	SendRRData Request over TCP (Fuzzed Session Handle)	94
Figure 71.	SendRRData Response over TCP (Fuzzed Session Handle).....	95
Figure 72.	SendRRData Request over TCP (Fuzzed Status)	95
Figure 73.	SendRRData Response over TCP (Fuzzed Status).....	96
Figure 74.	SendRRData Response over TCP (Fuzzed Sender Context).....	96
Figure 75.	SendRRData Request over TCP (Fuzzed Options)	97
Figure 76.	SendRRData Response over TCP (Fuzzed Options)	97
Figure 77.	SendRRData Response over TCP (Fuzzed Interface Handle).....	98
Figure 78.	SendRRData Request over TCP (Fuzzed Timeout).....	98
Figure 79.	SendRRData Response over TCP (Fuzzed Timeout)	99
Figure 80.	SendUnitData Request over TCP (Fuzzed Session Handle).....	99
Figure 81.	SendUnitData Response over TCP (Fuzzed Session Handle)	100

Figure 82.	SendUnitData Request over TCP (Fuzzed Status: 0x0000FFFF)	100
Figure 83.	SendUnitData Response over TCP (Fuzzed Status: 0x0000FFFF)	101
Figure 84.	SendUnitData Response over TCP (Fuzzed Sender Context)	101
Figure 85.	SendUnitData Request over TCP (Fuzzed Options).....	102
Figure 86.	SendUnitData Response over TCP (Fuzzed Options)	102
Figure 87.	SendUnitData Response over TCP (Fuzzed Interface Handle)	103
Figure 88.	SendUnitData Request over TCP (Fuzzed Timeout).....	103
Figure 89.	SendUnitData Response over TCP (Fuzzed Timeout)	104
Figure 90.	Reserved for Legacy Use Response over TCP (Fuzzed Command Field)	104
Figure 91.	Reserved for Legacy Use Response over UDP (Fuzzed Command Field)	105
Figure 92.	Reserved for Future Use Response over TCP (Fuzzed Command Field)	105
Figure 93.	Reserved for Future Use Response over UDP (Fuzzed Command Field)	106
Figure 94.	Get_Attributes_All Response over TCP (Class 0x01, Instance 0x01)	107
Figure 95.	Get_Attributes_All “Service Not Supported” Response over TCP (Class 0x06, Instance 0x01)	108
Figure 96.	Get_Attributes_All “Path Destination Unknown” Response over TCP (Class 0x28, Instance 0x01).....	108
Figure 97.	Get_Attributes_All “Path Destination Unknown” Response over TCP (Class 0x01, Instance 0x16).....	109
Figure 98.	Get_Attributes_All Response over TCP (Class 0x01, Instance 0x00)	109
Figure 99.	Get_Attribute_List Response over TCP (Class 0x01, Instance 0x01, Attribute 0x01).....	110
Figure 100.	Get_Attribute_List Response over TCP (Class 0x7F, Instance 0x01, Attribute 0x01).....	110

Figure 101.	Get_Attribute_List “Service Not Supported” Response over TCP (Class 0x01, Instance 0x01, Attribute 0x01)	111
Figure 102.	Get_Attribute_List “Path Destination Unknown” Response over TCP (Class 0x01, Instance 0x01, Attribute 0x01)	112
Figure 103.	Get_Attribute_List “Service Not Supported” Response over TCP (Class 0x01, Instance 0x01, Attribute 200)	113
Figure 104.	Get_Attribute_List Request over TCP (Attribute_count: 223).....	114
Figure 105.	Get_Attribute_List Response over TCP (Attribute_count: 223)	115
Figure 106.	Get_Attribute_List Request over TCP (Attribute_count: 224).....	115
Figure 107.	Get_Attribute_List Response over TCP (Attribute_count: 224)	116
Figure 108.	Get_Attribute_Single “Service Not Supported” Response over TCP	116
Figure 109.	Get_Attribute_Single “Path Destination Unknown” Response over TCP	117
Figure 110.	Get_Attribute_Single “Attribute Not Supported” Response over TCP	118
Figure 111.	Get_Attribute_Single “Service Not Supported” Response over TCP	118
Figure 112.	Get_Attribute_Single “Path Destination Unknown” Response over TCP	119
Figure 113.	Get_Attribute_Single “Attribute Not Supported” Response over TCP	119
Figure 114.	Find_Next_Object_Instance “Service Not Supported” Response over TCP	120
Figure 115.	Find_Next_Object_Instance “Path Destination Unknown” Response over TCP	121
Figure 116.	Find_Next_Object_Instance “Service Not Supported” Response over TCP	122
Figure 117.	Find_Next_Object_Instance “Path Destination Unknown” Response over TCP	122
Figure 118.	Find_Next_Object_Instance “Service Not Supported” Response over TCP	123
Figure 119.	Echo Response over TCP (0 Bytes Attached)	125

Figure 120.	Echo Response over TCP (243 Bytes Attached)	126
Figure 121.	Echo Response over TCP (8 Bytes Fuzzed)	126
Figure 122.	Echo Response over TCP (9 Bytes Fuzzed)	127
Figure 123.	Echo Response over TCP (10 Bytes Fuzzed)	127
Figure 124.	Echo Response over TCP (40 Bytes Fuzzed)	128
Figure 125.	Echo Response over TCP (243 Bytes Fuzzed)	128
Figure 126.	Echo Response over TCP (Maximum Number of Bytes without Error Message: 247 Bytes Fuzzed)	129
Figure 127.	Echo Response over TCP (248 Bytes Fuzzed)	129
Figure 128.	Echo Response over TCP (256 Bytes Fuzzed)	130
Figure 129.	Protected Typed File Read Response over TCP (Size Fuzzed)	130
Figure 130.	Protected Typed File Read Response over TCP (Tag Fuzzed)	131
Figure 131.	Protected Typed File Read Response over TCP (Offset Fuzzed)	131
Figure 132.	Protected Typed File Read Response over TCP (File Type Fuzzed)	132
Figure 133.	Protected Typed File Write Response over TCP (Size Fuzzed)	132
Figure 134.	Protected Typed File Write Response over TCP (Tag Fuzzed)	133
Figure 135.	Protected Typed File Write Response over TCP (Offset Fuzzed)	133
Figure 136.	Protected Typed File Write Response over TCP (File Type Fuzzed)	134
Figure 137.	Protected Typed File Write Response over TCP (Data Fuzzed)	134
Figure 138.	Protected Logical Write with Three Address Fields Response over TCP (Byte Size 0x00)	135
Figure 139.	Protected Logical Write with Three Address Fields Response over TCP (Byte Size Fuzzed)	135
Figure 140.	Protected Logical Write with Three Address Fields Response over TCP (File No. Fuzzed)	136
Figure 141.	Protected Logical Write with Three Address Fields Response over TCP (File Type Fuzzed)	136

Figure 142.	Protected Logical Write with Three Address Fields Response over TCP (Element No. Fuzzed).....	137
Figure 143.	Protected Logical Write with Three Address Fields Response over TCP (Sub-Element No. Fuzzed)	137
Figure 144.	Unprotected Read Response over TCP (Address Fuzzed)	138
Figure 145.	Unprotected Read Response over TCP (Size Fuzzed).....	138
Figure 146.	Diagnostic Status Response over TCP (Functionality Test).....	139
Figure 147.	Read Diagnostic Counters Response over TCP (Address Fuzzed: 0x3455)	139
Figure 148.	Read Diagnostic Counters Response over TCP (Size Fuzzed: 25).....	140
Figure 149.	Read Diagnostic Counters Response over TCP (Size Fuzzed: 75).....	140
Figure 150.	Restart Response over TCP (Functionality Test).....	141
Figure 151.	Download Completed Response over TCP (Functionality Test).....	141
Figure 152.	MicroLogix Protected Logical Read with Three Address Fields Request over TCP (File No. 0x03 and File Type 0x47)	142
Figure 153.	ControlLogix Protected Logical Read with Three Address Fields Request over TCP (File No. 0x03 and File Type 0x47)	143
Figure 154.	ControlLogix Protected Logical Read with Three Address Fields Response over TCP (File No. 0x03 and File Type 0x47).....	143

LIST OF TABLES

Table 1.	EtherNet/IP Packet Structure. Source: [16].	8
Table 2.	Message Format for Execute PCCC. Source: [26].	9
Table 3.	ENIP Test Commands. Source: [16].	14
Table 4.	CIP Test Commands. Source: [15].	15
Table 5.	PCCC Test Commands. Source: [17].	16
Table 6.	ENIP Test Plan.	20
Table 7.	CIP Test Plan	33
Table 8.	PCCC Testing Plan	40
Table 9.	ENIP Fuzz Testing Results	61
Table 10.	CIP Fuzz Testing Results	69
Table 11.	Identity Object Instance Attributes. Adapted from [15].	71
Table 12.	Identity Object Get_Attributes_All Response for Instance 0x00. Source: [15].	72
Table 13.	PCCC Fuzz Testing Results	76
Table 14.	Example MicroLogix 1100 Responses to PCCC Protected Logical Read with Three Address Fields Command	81
Table 15.	Summary of MicroLogix 1100 Unexpected Responses	82
Table 16.	Summary of ControlLogix 1756-L71 Response Deviations	84

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

AB/RA	Allen Bradley/Rockwell Automation
CIP	Common Industrial Protocol
DNP3	Distributed Network Protocol
DoS	denial of service
EtherNet/IP	EtherNet Industrial Protocol
ENIP	EtherNet/IP
EXT STS	Extended Status
HM&E	hull mechanical and electrical
ICS	Industrial Control System
IOI	Internal Object Identifier
NOP	No Operation
ODVA	Open DeviceNet Vendor Association
PCCC	Programmable Controller Communication Commands
PROM	programmable read-only memory
PLC	programmable logic controller
RAM	random access memory
TCP	Transmission Control Protocol
SCADA	supervisory control and data acquisition
STS	Status
UDP	User Datagram Protocol

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would like to thank my thesis advisors, Thuy Nguyen and Dr. Cynthia Irvine, for their guidance, expertise, and encouragement. You challenged me and drove me to make the most of this opportunity. I would also like to thank Francisco Tacliad, whose thesis laid the foundation for our research, and David Shifflet, for his assistance in creating our testing environment. To my wife, Ashley, thank you for your unwavering support. Your love, patience, and encouragement were invaluable during this process. To my daughter, Grace, and newborn son, James, you bring me laughter and joy every day and have been a source of great motivation. Thank you also to my parents and sister who continually support me, not just during this period, but throughout my life. Lastly, thank you to the Naval Postgraduate School professors, from whom I have learned so much over these past two years.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. MOTIVATION

While industrial control systems (ICSs) allow for the management of large, complex, and often distributed machinery systems, they can also be manipulated for malicious purposes. In 2000, a disgruntled former employee in Queensland, Australia, perpetrated one of the first known attacks on a Supervisory Control and Data Acquisition (SCADA) system. Through manipulation of the pumping stations, the offender released over one million liters of sewage into local waterways [1]. Following the events in Australia, attackers have successfully exploited the vulnerabilities inherent in networked control systems. The Stuxnet worm, discovered in 2010, targeted specific Siemens programmable logic controllers (PLCs) used at the Natanz nuclear enrichment facility. The sophisticated malware utilized four zero-day vulnerabilities to send fatigue-inducing commands to PLCs controlling nuclear enrichment centrifuges [2].

SCADA-based power grids are also vulnerable to cyber attacks. In December 2015, the Prykarpattyaoblenergo control center in Ukraine was the victim of an attack that left more than 230,000 West Ukraine residents without power for six hours. The alleged Russian attackers gained access to the utility's network through a phishing scheme. Using a program called BlackEnergy3, the hackers established a backdoor on the network, from which they gained access to the SCADA networks. The attackers were able to take thirty substations offline, disable backup power, and rewrite substation firmware before using KillDisk malware to delete files from operator systems and render them unusable [3].

Attacks against networked control systems can take varied forms. In April 2017, hackers simultaneously set off all 156 tornado warning sirens in Dallas, Texas. In normal operation, police dispatchers or weather officials send signals to a transmitter that activates selected sirens. To set off the sirens, the attacker used the input frequency to repeatedly activate all of the sirens over a period of several hours [4]. Although this

attack relied on hijacking radio frequencies, similar disruption could potentially be caused by malicious software.

Efforts to address PLC vulnerabilities started several years ago. In 2007, the Idaho National Laboratory conducted the Aurora test in which researchers caused physical damage to a diesel generator by rapidly connecting and disconnecting the generator to the power grid, causing an out of phase condition [5]. In 2013, Sandia National Laboratories developed a system called Weaselboard, which provides zero-day exploit protection for PLCs by monitoring PLC backplane communications between devices and scanning for configuration changes [6]. In the private sector, Digital Bond created Project Basecamp to perform security testing on popular SCADA system components. The Project Basecamp researchers demonstrated vulnerabilities affecting multiple different PLC market leaders [7].

One effective method to test for vulnerabilities in protocols and systems is *fuzzing* or *fuzz testing*. Fuzzing is a technique that aims to uncover coding errors or security flaws by feeding a target program random input parameters [8]. Previous work has demonstrated that fuzz testing can be used to uncover vulnerabilities in industrial network protocol [9].

B. RELEVANCE TO THE NAVY

The Hull Mechanical and Electrical (HM&E) systems on U.S. Navy ships employ industrial automation components such as PLCs to run critical onboard services like propulsion, auxiliary, and mission-specific equipment [10]. As the Navy reduces shipboard crew strength through automation, as demonstrated in the DDG 1000, and launches completely unmanned vessels like the Anti-Submarine Warfare Continuous Trail Unmanned Vessel, the reliance on shipboard ICS increases. While a networked control system architecture (e.g., SCADA) provides centralized data availability and control of physical equipment in different locations, the communication channels between the PLCs and control devices are vulnerable to cyber attacks [2], [11].

Recognizing the inherent vulnerabilities in control systems, the Navy is developing the Resilient Hull, Mechanical, and Electrical Security system to prevent

attackers from disabling or accessing shipboard PLCs. The system varies the implementation of PLC firmware so that if an exploit is able to disable a primary controller, the same exploit will not affect the redundant PLC's ability to assume the operation [12].

Allen Bradley / Rockwell Automation (AB/RA) is a leader in the ICS field and their products are currently used onboard Navy ships. Grandgenett et al. showed that AB/RA PLCs are susceptible to denial-of-service (DoS) [13], man-in-the-middle attacks, and replay attacks to force unauthorized privileged commands [14]. AB/RA PLCs support two widely-used industrial control protocols: Common Industrial Protocol (CIP) [15], EtherNet/IP (ENIP) [16], in addition to Programmable Controller Communication Commands (PCCC), a legacy AB/RA proprietary protocol [17]. CIP is an industry-vetted network protocol used to manage industrial devices [15]. CIP rides on top of ENIP, which is transported over TCP/IP. ICS network protocols, like CIP, ENIP, and PCCC allow for efficient control of distributed systems, but also create potential vectors of attack to disable or destroy U.S. Navy ships.

C. OBJECTIVES

This thesis aims to identify vulnerabilities in select AB/RA PLCs through their implementation of CIP and ENIP to directly improve mission readiness of U.S. Navy ships and harden their cyber defenses. Tacliad discusses the discovery of a CIP-encapsulated PCCC vulnerability in an AB/RA MicroLogix PLC through fuzz testing different ENIP, CIP, and PCCC commands [9]. This thesis seeks to expand and improve the ENIP Fuzz program to include additional ENIP, CIP, and PCCC commands. Once adapted to fuzz a larger catalogue of commands, we aim to implement ENIP Fuzz on the MicroLogix PLC and a more advanced AB/RA PLC (ControlLogix) to determine if vulnerabilities to AB/RA PLC communications stack are cross-generational.

D. THESIS ORGANIZATION

Chapter II provides background on CIP, EtherNet/IP, and PCCC protocols. It includes a summary of previous ICS fuzz testing efforts, Scapy [18], and existing Scapy-based fuzzing tools. Chapter II also presents an introduction to two AB/RA PLCs used in

this thesis. Chapter III describes the experimentation design objectives, methodology, and testing environment. Chapter IV is an account of test plan and implementation. Chapter V is our analysis of results. Chapter VI discusses conclusions and future work.

II. BACKGROUND

A. COMMON INDUSTRIAL PROTOCOL (CIP)

CIP, previously known as Control and Information Protocol [19], is a “peer-to-peer object oriented protocol that provides connections between industrial devices (sensors, actuators) and higher-level devices (controllers)” [15]. CIP was developed by Rockwell Automation but is now run by Open DeviceNet Vendors Association (ODVA), a global association of automation industry leaders. CIP is supported by four different ODVA network communication protocols, EtherNet/IP, DeviceNet, CompoNet and ControlNet [20]. Using the Open System Interconnection model, CIP utilizes the Presentation and Application layers. Session layer is not utilized in CIP. In the EtherNet/IP structure, CIP rides on top of the Transport layer and utilizes an Ethernet network stack [21]. Figure 1 illustrates the CIP network work stack architecture.

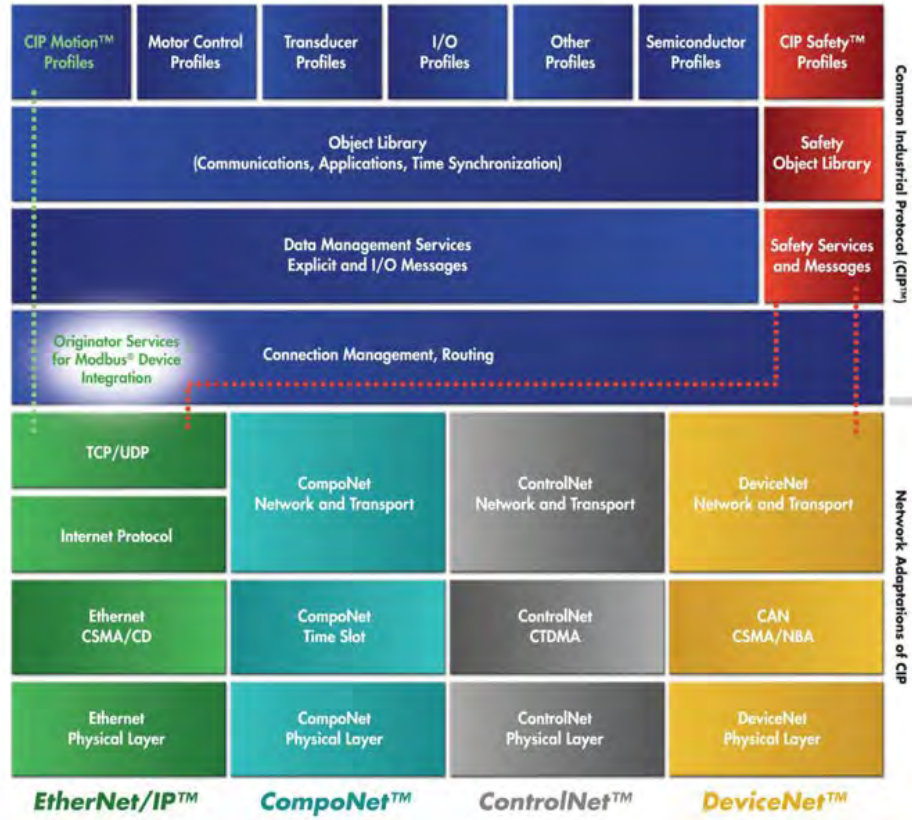


Figure 1. CIP Network Architecture Stack. Source: [15].

CIP nodes are comprised of *objects*, which can contain data. Each object is an *instance* of a particular *class*. CIP objects contain attributes for both object and class, which enable specific services. Objects with the same attributes belong to the same class [15]. CIP is designed so that the same objects on different devices behave in the same manner. This allows for a producer-consumer relationship, where data is sent from the producer device to potentially multiple consumer devices with a single transmission [22]. Figure 2 shows the CIP Object Model.

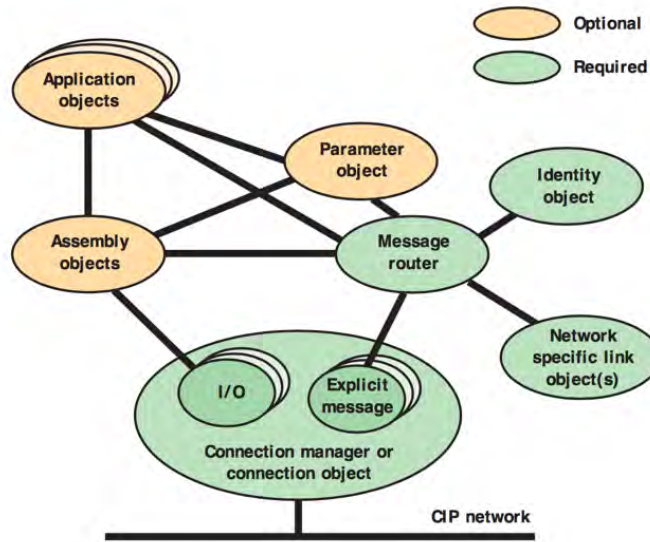


Figure 2. CIP Object Model. Source: [23].

CIP relies on two methods of routing to transmit data. For connected messages, CIP uses a connection ID to transfer packets. For unconnected messages, an Internal Object Identifier (IOI), also known as an EPATH, is used to explicitly provide the path packets will travel to their destination. The device that opens the connection dictates the routing directives [24].

B. ETHERNET/INDUSTRIAL PROTOCOL (ETHERNET/IP)

EtherNet/IP utilizes Ethernet (IEEE 802.3) and the TCP/IP network protocol stack to transport CIP as an application layer protocol. For this reason, it is often referred to as “CIP over Ethernet” [21]. EtherNet/IP uses IP Multicast to enable a producer-consumer exchange of information between a sending device and receiving devices [15]. By utilizing a common Ethernet protocol stack, EtherNet/IP allows CIP to be used across different CIP networks and enables Internet compatibility and remote control capability [21]. Figure 3 shows how an EtherNet/IP message is embedded in the TCP data payload.

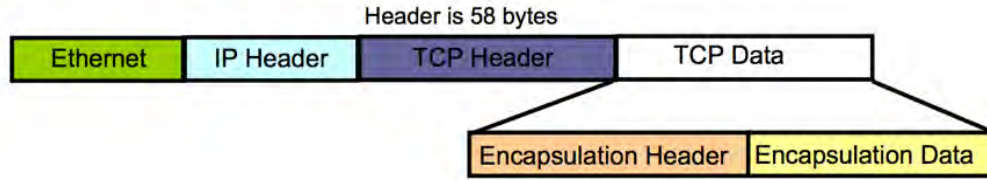


Figure 3. EtherNet/IP Packet Encapsulation. Source: [25].

The encapsulation message includes a standard 24-byte fixed length header, followed by an optional data section. Encapsulation messages may be in TCP or UDP format and are sent to port 44818 of the receiving device. Table 1 shows the content of the EtherNet/IP encapsulation header and encapsulated data [16].

Table 1. EtherNet/IP Packet Structure. Source: [16].

Structure	Field Name	Data Type	Field Value
Encapsulation header	Command	UINT	Encapsulation command
	Length	UINT	Length, in bytes, of the data portion of the message, i.e., the number of bytes following the header
	Session handle	UDINT	Session identification (application dependent)
	Status	UDINT	Status code
	Sender Context	ARRAY of octet	Information pertinent only to the sender of an encapsulation command. Length of 8.
	Options	UDINT	Options flags
Command specific data	Encapsulated data	ARRAY of 0 to 65511 octet	The encapsulation data portion of the message is required only for certain commands

C. PROGRAMMABLE CONTROLLER COMMUNICATION COMMANDS (PCCC)

PCCC is a legacy AB/RA protocol designed for the PLC5 and SLC500 processors [21]. PCCC objects do not support CIP connections on their own. However, they can be encapsulated in CIP commands in order to communicate with legacy PLCs. This encapsulation is accomplished through the use of an IOI. Once a connection to a Message Router object is established, an IOI is used to specify the PCCC object. When the CIP packet is received, “Execute PCCC” service is processed by the PCCC object at the

receiving device [24]. Table 2 shows the message structure for a PCCC command, without CIP encapsulation [26].

Table 2. Message Format for Execute PCCC. Source: [26].

Request			Response		
Name	Data Type	Description	Name	Data Type	Description
Length	USINT	Length of requestor ID	Length	USINT	Length of requestor ID
Vendor	UINT	Vendor number of requestor	Vendor	UINT	Vendor number of requestor
Serial Number	UDINT	ASA serial number of requestor	Serial Number	UDINT	ASA serial number of requestor
Other	Product Specific	Identifier of user, task, etc. on the requestor	Other	Product Specific	Identifier of user, task, etc. on the requestor
CMD	USINT	Command byte	CMD	USINT	Command byte
STS	USINT	0	STS	USINT	Status byte
TNSW	UINT	Transport word	TNSW	UINT	Transport word. Same value as the request.
FNC	USINT	Function code. Not used for all CMD's.	EXT_STS	USINT	Extended status. Not used for all CMD's.
PCCC_params	ARRAY of USINT	CMD/FNC specific parameters	PCCC_results	ARRAY of USINT	CMD/FNC specific result data

D. FUZZ TESTING

The field of fuzz testing originated with Wisconsin University professor Barton Miller in 1989. Miller's team built a program, named *fuzz*, which generated random strings of characters and fed them into program inputs in an effort to create system failures [8]. Fuzz testing has grown into a widely-used method of vulnerability testing.

There are two main subcategories of fuzzers: generation-based and mutation. Generation-based fuzzers craft fuzzing inputs based on knowledge of input structures and protocols. These programs generate strings of random characters and varying lengths. Sophisticated generation-based fuzzers utilize block-based methods, where each input field is treated as a targetable fuzzing block [27]. These fuzzers require detailed specifications of input fields and protocols in order to customize block-sized inputs [28].

Mutation fuzzers utilize known good inputs and network traffic to build fuzzing structures. By taking the known good input and switching out acceptable values with random values, mutation fuzzers increase the likelihood their malformed inputs will not be rejected outright, which increases their effectiveness [27].

E. ICS FUZZERS

Numerous fuzz testing suites targeting well known ICS protocols are available. beSTORM offers a commercially available EtherNet/IP fuzzing tool [29]. Mu Test Suite, also a commercial product, includes resources to fuzz Distributed Network Protocol (DNP3), Modbus, and the IEC61850 protocol [27]. In the open source arena, the Sulley fuzzer includes modules for popular ICS protocols such as DNP3, Inter-Control Center Communications Protocol, and Modbus [30]. Developed at Dartmouth, LZFuzz fuzzes SCADA communications with unknown protocol structures. LZFuzz inserts itself into live traffic and captures packets. Packets inbound to the target are tokenized and sent through a mutation fuzzer to generate fuzzing inputs to the target. The program then monitors return traffic to the traffic source for indications of success [27].

This thesis research utilizes Tacliad's open source fuzzing tool, called ENIP Fuzz. ENIP Fuzz is an ICS fuzzing program that uses the Python-based packet manipulation tool, Scapy [18] to craft customized fuzzing inputs. ENIP Fuzz targets fields within ENIP and CIP request packets [9].

F. SCAPY

Scapy is a Python-based packet manipulation tool that can enable network probes and attacks. Scapy is flexible enough to allow custom packet crafting. It does not place limits on type of field input or stack configuration, which makes it a powerful tool for protocol fuzz testing. Users can craft Scapy packets in stackable layers. Scapy is capable of both sending and listening for response packets. Many networking tools apply interpretive filters on packet responses. Scapy does not employ this method in order to avoid inserting potential bias into response results. Interpretation of Scapy response packets lies with the user [18].

G. PREVIOUS SCAPY-BASED FUZZING

Scapy's versatile configuration has made it a popular choice for fuzz testing frameworks. Scapy allows a user to specify designated fields for fuzzing, while providing standard protocol inputs to other fields [31]. Scapy libraries have been used to fuzz Wi-Fi drivers [32], IPV6 [33] and IPV6 over low power wireless personal area networks [34],

and Internet Key Exchange messages [35]. In the ICS field, different fuzzing tools have utilized Scapy. Modbus/TCP Fuzzer targets the Modbus communication protocol [36]. Modbus is an application layer protocol that utilizes a master-slave architecture [37]. Scapy is used to target the Modbus/TCP master-initiated command packets for fuzzing. Some electrical utilities use the IEEE C37.118 protocol to communicate between wide area monitoring systems that operate phasor measurement units and phasor data concentrators. Sprabery et al. created a IEEE C37.118 mutation-based fuzzer using Scapy to test particular protocol rules for vulnerabilities [38].

Tacliad's ENIP Fuzz targets the EtherNet/IP and CIP protocols using the Scapy library to craft malformed packets. ENIP Fuzz tests specified objects in the designated protocols and monitors for unexpected responses or lack of response to liveliness checks. While Tacliad tested a very limited sample of EtherNet/IP, CIP, and CIP-encapsulated PCCC commands, his experimentation demonstrated a proof of concept, which can be greatly expanded to determine the robustness of the examined protocols [9].

H. ALLEN-BRADLEY / ROCKWELL AUTOMATION PLCS (MICROLOGIX 1100 AND CONGROLLOGIX 5570)

The MicroLogix 1100 is a lower-end PLC that supports 12 inputs (10 digital and 2 analog) and 6 outputs, and up to 144 digital I/O points. It is utilized to perform varied industrial applications such as machinery control and production processes. The controller has an RS232/485 serial port and an Ethernet port. The Ethernet port enables peer-to-peer communication across controllers [39]. Figure 4 shows a MicroLogix 1100.



Figure 4. MicroLogix 1100 PLC. Source: [39].

The AB/RA ControlLogix PLC is a more advanced modular PLC than the MicroLogix 1100. A ControlLogix PLC consists of a controller (CPU) module (e.g., 1756-L71 controller) and multiple I/O modules in one chassis. The local I/O modules can include one or more EtherNet/IP modules (e.g., 1756-EN2T and 1756-EWEB modules), and one or more analog and digital I/O modules (e.g., 1756-OF8 and 1756-IB16 modules). A ControlLogix 5570 PLC can handle up to 128,000 digital or 4,000 analog I/O points and is used for shipboard applications, power generation, and transportation functions. The PLC can communicate across multiple protocols including EtherNet/IP (including CIP and encapsulated PCCC), ControlNet, DeviceNet, Data Highway Plus, Remote I/O, SynchLink, and third-party networks. The 5570 model does not offer an embedded Ethernet Port, but has a USB interface for local programming. For ease of configuration and maintenance, most EtherNet/IP modules support web browsing, email, and file transfer. The ControlLogix family also offers the ability to configure controller redundancy into the system. [40]. Figure 5 shows a ControlLogix PLC with multiple I/O modules.



Figure 5. ControlLogix PLC. Source: [40].

III. DESIGN

A. OBJECTIVES

This thesis explores two objectives. The first objective is to determine if ENIP Fuzz can be used to determine new vulnerabilities in the AB/RA implementation of the ENIP, CIP and PCCC protocols used by the MicroLogix and ControlLogix PLCs. Our hypothesis is that undiscovered software flaws could potentially exist in the implementation of AB/RA’s implementation of the protocols. The second objective is to determine if testing network vulnerabilities known to exist in older PLCs help inform on the robustness of the ICS network stack in a more modern PLC design. Our hypothesis is that legacy protocol handlers are left in the code base but not fully tested in newer PLC models.

B. METHODOLOGY

Testing follows a black box-style fuzzing methodology, i.e., having no access to AB/RA source code. The test plan and testing methodology relies heavily on the protocol specifications for ENIP, CIP, and PCCC protocols. To determine specific commands from each protocol to fuzz, we analyze protocol commands to identify targets that focus on non-disruptive functionality. We avoid commands that we assessed to have high risk of reconfiguring memory, altering functionality, or causing permanent damage to the SUT. We aim to select target commands that provide a representative sample of different types of services provided by each protocol.

Previous testing using ENIP Fuzz exercised three MicroLogix-supported commands sent over a TCP connection: ENIP Register Session, CIP No_Operation (NOP), and PCCC Execute Services [9]. Our testing framework focuses on a wider cross-section of ENIP commands and CIP services, transported over both TCP and UDP, in an effort to discover vulnerabilities that may be present in different service types.

The ENIP test commands can be grouped into five categories as shown in Table 3. Our ENIP test suite consists of all three “list” commands, the UnRegisterSession

command, the SendRRData and SendUnitData commands, the reserved for legacy commands, and the reserved for future expansion commands.

Table 3. ENIP Test Commands. Source: [16].

ENIP Test Commands	Description
Lists	
List Identity	Requests information on the target's identity.
List Interfaces	Requests non-CIP communication interfaces associated with the target.
List Services	Requests information on the supported services.
Session Commands	
Unregister Session	Instructs the receiver to initiate a close of the underlying TCP/IP connection.
Send Commands	
SendRRData	Transfers an encapsulated request/reply packet.
SendUnitData	Sends encapsulated connected messages.
Legacy Commands	
Reserved Command Codes	Reserved for legacy use.
Future Expansion Commands	
Reserved Command Codes	Reserved for future expansion.

For the CIP Explicit Messaging testing, we select services with multiple fuzzable fields based on the assumption that such commands would be more complex and have a higher potential for vulnerabilities in handling errors. Table 4 summarizes the CIP common services in the CIP test suite. While each of the Get_Attributes_xxx services have a corresponding Set_Attributes_xxx command, we specifically skip the latter in an effort to not corrupt any PLC settings.

Table 4. CIP Test Commands. Source: [15].

CIP Test Commands	Description
Get Attribute All	Returns the contents of the instance or class attributes defined in the object definition.
Get Attribute List	Returns the contents of the selected gettable attributes of the specified object class or instance.
Get Attribute Single	Returns the contents of the specified attribute.
Find Next Object Instance	Returns a list of Instance IDs [15] associated with existing Object Instances [15]. Existing Objects are those that are currently accessible from the CIP subnet.

Our strategy for testing PCCC commands follows two common testing techniques: specification compliance testing and unexpected exception handling testing. First, we identify the PCCC commands that are described in the DF1 Protocol and Command Set specification [17] as compatible with the MicroLogix 1000 family's implementation of the protocol. PCCC information for the SUTs is not publicly available. Table 5 shows the commands in the PCCC test suite that have a low risk of disrupting the SUT functionality. We choose the PCCC Echo command because it allows the inclusion of a large amount of data in a packet, which can be used to test the maximum allowable packet size. We select the Protected Typed File Read, Protected Typed File Write, and Protected Logical Write with Three Address Fields commands for their multiple fuzzable fields and potential for stack corruption. The Unprotected Read command is selected for its potential to cause errors by attempting to read unintended address spaces. The Read Diagnostic Counters command is included in the test suite due to its ability to read data from a fuzzable address location. The Diagnostic Status command is also tested because the response to the Diagnostic Status request command provides the starting memory address for the PLC's diagnostic counters, which can be used with the Read Diagnostic Counters command.

In addition to the MicroLogix-supported PCCC commands, the PCCC test suite also includes commands that may contain vulnerabilities or cause an unexpected result

because, according to the PCCC specification [17], they are not supported by the MicroLogix 1000 PLC (see Table 5). While the selected commands, Download Completed and Restart, do not have fuzzable fields, their inclusion in the test suite allows testing of unexpected error handling.

Table 5. PCCC Test Commands. Source: [17].

PCCC Test Commands	Description
Echo	The receiving module should reply to this command by transmitting the same data back to the originating node.
Protected Typed File Read	Reads data from an open file in the PLC.
Protected Typed File Write	Writes data to an open file in the PLC.
Protected Logical Write with Three Address Fields	Writes data to a logical address in PLC processor.
Unprotected Read	Read data from a common interface file.
Diagnostic Status	Reads a block of status information from an interface module.
Read Diagnostic Counters	Reads up to 244 bytes of data from the PROM or RAM of an interface module.
Restart	Revokes upload and download privileges for the source computer node and initializes PLC restart. (Command intended for PLC-3 only after completion of upload or download operation)
Download Completed	Places processor back in previous mode upon completion of system download.
Protected Typed Logical Read with Three Address Fields	Reads data from a logical address in PLC processor.

Previous ENIP Fuzz testing uncovered an improper input validation vulnerability in different versions of MicroLogix 1100 controllers, which is described in the ICS-CERT security advisory ICSA-17-138-03 [41]. When the Protected Typed Logical Read with Three Address Fields command was issued with certain parameters, the MicroLogix 1100 halted, causing a denial of service condition. This command is tested on a ControlLogix 5570 to verify our second hypothesis that legacy protocol handlers may be left in the code base but not fully tested in newer PLC models.

C. TEST ENVIRONMENT

The fuzzing tool used in this thesis is ENIP Fuzz. It is a Scapy-based fuzzer that enables construction of specially crafted packets, which allows the user to test a wide variety of inputs for each value in protocol packet. ENIP Fuzz utilizes both CIP and ENIP dissectors, which define classes for each protocol request and response message format.

The MicroLogix test environment consists of a MicroLogix system under test (SUT), a Windows PC with a Windows 7 virtual machine (VM), a Mac laptop with a Kali Linux 2.0 VM, and a Mac laptop running the Wireshark protocol analyzer. All components are connected to a central hub. The Windows 7 VM runs RSLinx and RSLogix—AB/RA development software with which a user can send commands to and monitor responses from the connected PLC. In the Kali VM, ENIP Fuzz is used to build and send custom packets to the PCL in order to test the ENIP, CIP, and PCCC protocols for vulnerabilities. During testing, potential faults are monitored on the RSLogix console, from fault responses in Wireshark, and physical fault indications on the SUT. The testing environment setup is displayed in Figure 6.

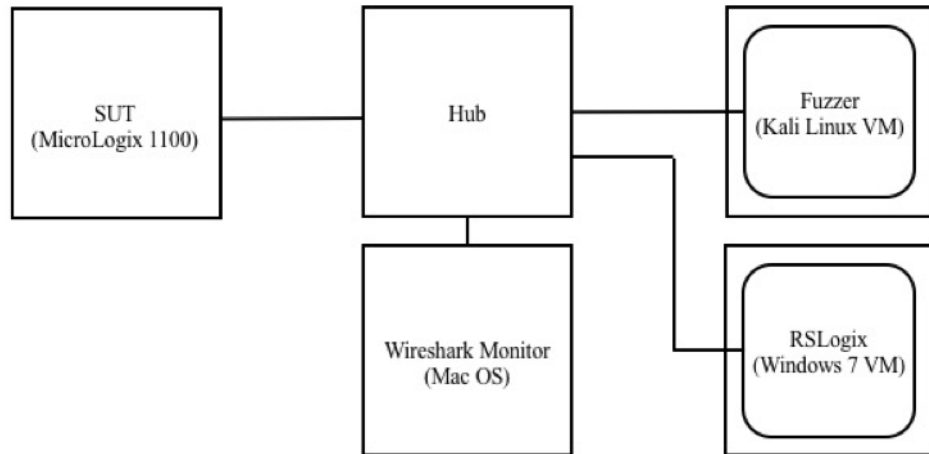


Figure 6. MicroLogix Testing Environment

The ControlLogix test environment is similar to the MicroLogix environment except that the Rockwell Studio 5000 Logic Designer development software running on a Window 7 PC is used instead of the RSLogix software (see Figure 7).

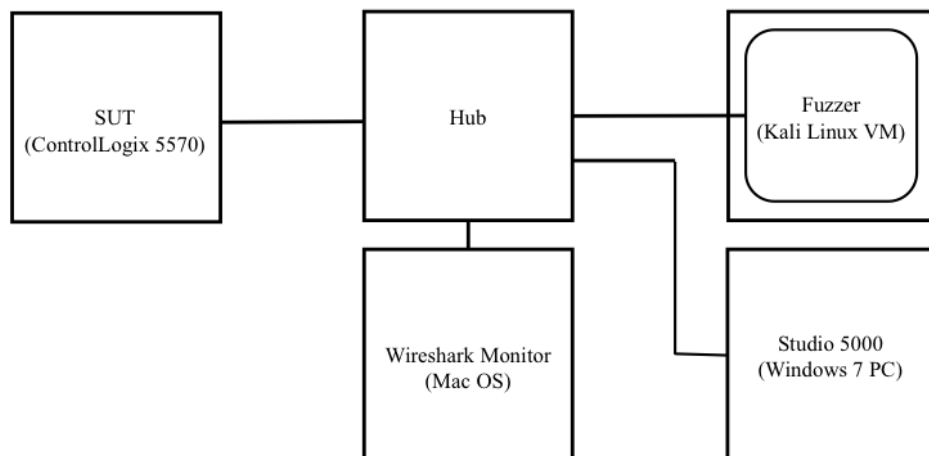


Figure 7. ControlLogix Testing Environment

IV. IMPLEMENTATION AND TEST PLAN

A. FUZZER IMPLEMENTATION

The fuzzing platform, ENIP Fuzz [9], is modified to conduct the desired breadth of target command testing across the ENIP, CIP, and PCCC protocols. Using the modified ENIP Fuzz program, properly formed packets are crafted and sent to the SUT to establish baseline request and response behavior. Specially designed malformed packets are then sent to the SUT and analyzed in relation to the hypothesized SUT responses. The testing goal is to trigger a denial of service condition in the SUT. This is defined as a fault in the SUT that requires either a power cycle to clear or reset through the RSLogix/Studio 5000 interfaces, or a disruption in the SUT's ability to send or receive command traffic.

1. FUZZER MODIFICATIONS FOR MICROLOGIX

The ENIP Fuzz architecture consists of command and service-specific fuzzing modules and protocol dissectors. Eight ENIP fuzzing modules are constructed to test the following ENIP commands (discussed in Chapter III): ListServices, ListIdentity, ListInterfaces, UnRegisterSession, SendRRData, SendUnitData, Reserved for Legacy Use, and Reserved for Future Use. Two CIP fuzzing modules are created to test the Get_Attributes_All and Find_Next_Object_Instance CIP services. Nine PCCC fuzzing modules are added to test the following PCCC commands via the PCCC Execute Service Request service: Echo, Protected Typed File Read, Protected Typed File Write, Protected Typed Logical Write with Three Address Fields, Unprotected Read, Download Completed, Restart, Diagnostic Status, and Read Diagnostic Counters.

The ENIP Fuzz CIP dissector is modified to allow the Find_Next_Object_Instance command to specify the number of maximum values returned. Both ENIP and CIP dissectors are modified to create the expanded packet views presented later in this document.

2. FUZZER MODIFICATIONS FOR CONTROLLOGIX

In order to test a recently discovered PCCC vulnerability [9] affecting MicroLogix on the ControlLogix PLC, ENIP Fuzz's handling of the Protected Typed Logical Read with Three Address Fields PCCC command requires modifications. The objective of this test is to determine whether the PCCC vulnerability in the MicroLogix implementation also exists in the ControlLogix software. Through analysis of ControlLogix network traffic, it is observed that the ControlLogix implements the CIP Forward_Open request differently. The Forward_Open request establishes a connection with a target device [15] and precedes the target test command request. ControlLogix PLCs require a 3-word request path [15], as opposed to the 2-word request path used on the MicroLogix. The request path specifies the required route the command packet travels to the remote target device [15]. ENIP Fuzz is modified to handle both types of request path.

B. ENIP FUZZING TEST PLAN

Previous ENIP Fuzz testing is limited to the RegisterSession command [9]. The current work expands the testing to test ENIP commands not tested by Tacliad [9] for vulnerabilities. Command fields are tested in isolation in order to provide a methodical evaluation of each command's potential vulnerabilities. Table 6 summarizes the ENIP test plan.

Table 6. ENIP Test Plan

Test Number	ENIP Command	Fuzzed Field	Protocol	Fuzzing Parameters
T1	List Services/Identity/Interfaces	Session Handle	TCP	0x00000000 to 0xFFFFFFFF
T2	List Services/Identity/Interfaces	Session Handle	UDP	0x00000000 to 0xFFFFFFFF
T3	List Services/Identity/Interfaces	Status	TCP	0x00000000 to 0xFFFFFFFF
T4	List Services/Identity/Interfaces	Status	UDP	0x00000000 to 0xFFFFFFFF

Test Number	ENIP Command	Fuzzed Field	Protocol	Fuzzing Parameters
T5	List Services/Identity/Interfaces	Sender Context	TCP	0x0000000000000000 to 0xFFFFFFFFFFFFFFFF
T6	List Services/Identity/Interfaces	Sender Context	UDP	0x0000000000000000 to 0xFFFFFFFFFFFFFFFF
T7	List Services/Identity/Interfaces	Options	TCP	0x00000000 and 0xFFFFFFFF
T8	List Services/Identity/Interfaces	Options	UDP	0x00000000 and 0xFFFFFFFF
T9	UnRegisterSession	Session Handle	TCP	0x00000000 to 0xFFFFFFFF
T10	UnRegisterSession	Status	TCP	0x00000000 to 0xFFFFFFFF
T11	UnRegisterSession	Sender Context	TCP	0x0000000000000000 to 0xFFFFFFFFFFFFFFFF
T12	UnRegisterSession	Options	TCP	0x00000000 and 0xFFFFFFFF
T13	UnRegisterSession UDP Functionality	N/A	UDP	Properly crafted ENIP encapsulated packet sent over UDP
T14	SendRRData	Session Handle	TCP	0x00000000 to 0xFFFFFFFF
T15	SendRRData	Status	TCP	0x00000000 to 0xFFFFFFFF
T16	SendRRData	Sender Context	TCP	0x0000000000000000 to 0xFFFFFFFFFFFFFFFF
T17	SendRRData	Options	TCP	0x00000000 and 0xFFFFFFFF
T18	SendRRData	Interface Handle	TCP	0x00000000 and 0xFFFFFFFF
T19	SendRRData	TimeOut	TCP	0-65535
T20	SendUnitData	Session Handle	TCP	0x00000000 to 0xFFFFFFFF
T21	SendUnitData	Status	TCP	0x00000000 to 0xFFFFFFFF
T22	SendUnitData	Sender Context	TCP	0x0000000000000000 to 0xFFFFFFFFFFFFFFFF
T23	SendUnitData	Options	TCP	0x00000000 and 0xFFFFFFFF
T24	SendUnitData	Interface Handle	TCP	0x00000000 and 0xFFFFFFFF

Test Number	ENIP Command	Fuzzed Field	Protocol	Fuzzing Parameters
T25	SendUnitData	TimeOut	TCP	0-65535
T26	Reserved for Legacy	Command Field	TCP	0x0001,0x0002, 0x0005, 0x0067-0x006E, and 0x0071-0x00C7
T27	Reserved for Legacy	Command Field	UDP	0x0001,0x0002, 0x0005, 0x0067-0x006E, and 0x0071-0x00C7
T28	Reserved for Future Use	Command Field	TCP	0x0006-0x0062 and 0x00C8-0xFFFF
T29	Reserved for Future Use	Command Field	UDP	0x0006-0x0062 and 0x00C8-0xFFFF

1. ENIP ListServices Command

The ENIP ListServices Request command returns the service(s) the target supports. To test the command, the Session Handle, Status, Sender Context, and Options fields are individually fuzzed using both TCP and UDP. The Session Handle field is tested with a combination of inputs ranging from 0x00000000 to 0xFFFFFFFF. The Status field is fuzzed in a similar manner. The Sender Context field is tested with data ranging from 0x0000000000000000 to 0xFFFFFFFFFFFFFFFF. The Options field is tested between 0x00000000 and 0xFFFFFFFF. Figures 8–11 illustrate the packet structure for the ListServices command sent over TCP and UDP, respectively.

```

###[ ENIP TCP ]###
Command   = List Services (0x0004)
Length    = None
Session_Handle= 0x0
Status    = Success
Sender_Context= 0
Options   = 0

```

Fields encapsulated at the ENIP layer are highlighted.

Figure 8. An Example ENIP ListServices Request over TCP Packet

```

0000  45 00 00 40 00 01 00 00 40 06 f8 ed c0 a8 00 3e
0010  c0 a8 00 3b af 12 af 12 00 00 00 00 00 00 00 00
0020  50 02 20 00 ab db 00 00 04 00 00 00 00 00 00 00
0030  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

```

Figure 9. Hexadecimal View of Example ENIP ListServices Request over TCP Packet

```

###[ ENIP UDP ]###
Command   = List Services (0x0004)
Length    = None
Session_Handle= 0x0
Status    = Success
Sender_Context= 0
Options   = 0

```

Fields encapsulated at the ENIP layer are highlighted.

Figure 10. An Example ENIP ListServices Request over UDP Packet

```

0000  45 00 00 34 00 01 00 00 40 11 f8 ee c0 a8 00 3e
0010  c0 a8 00 3b f3 7a af 12 00 20 d7 56 04 00 00 00
0020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0030  00 00 00 00

```

Figure 11. Hexadecimal View of Example ENIP ListServices Request over UDP Packet

2. ENIP UnRegisterSession Command

The ENIP UnRegisterSession command terminates an existing ENIP session and closes the TCP connection associated with the particular ENIP session. An ENIP session is established using the ENIP RegisterSession command that was previously tested [9]. After receiving the UnRegisterSession command, the receiver initiates the closing of the TCP connection and does not reply with a response message. In the event this command is sent via UDP, the receiver replies with an error code 0x01, indicating an invalid or unsupported command [16]. The receiver always closes the TCP connection even if the

UnRegisterSession command contains unexpected values, e.g., invalid session handle [16].

To test the UnRegisterSession commands, the Session Handle, Status, Sender Context, and Options fields are fuzzed. The Session Handle field is tested with a combination of inputs ranging from 0x00000000 to 0xFFFFFFFF. The Status field is fuzzed in a similar manner. The Sender Context field is tested from 0x0000000000000000 to 0xFFFFFFFFFFFFFFFF. The Options field is tested with values between 0x00000000 and 0xFFFFFFFF. Aside from the Session Handle test, the other fields are fuzzed using a valid Session Handle in the packet.

To determine if MicroLogix complies with the ENIP requirement that an UnRegisterSession command sent over UDP will be rejected with an error code of 0x01 “invalid or unsupported command” [16], a single properly-crafted UDP UnRegisterSession command is included in the ENIP test suite. Figures 12 and 13 display a sample TCP ENIP UnRegisterSession Request. Figures 14 and 15 show a UDP version of the command for exception testing purposes.

```
###[ ENIP TCP ]###  
    Command    = UnRegister Session (0x0066)  
    Length     = None  
    Session_Handle = 0x0  
    Status      = Success  
    Sender_Context = 0  
    Options     = 0  
###[ Unregister Session ]###
```

Fields encapsulated at the ENIP layer are highlighted.

Figure 12. An Example ENIP UnRegisterSession Request over TCP Packet

```

0000  00 1d 9c ca cb 1b 90 e2 ba 18 fc 2e 08 00 45 00
0010  00 40 00 01 00 00 40 06 f8 ed c0 a8 00 3e c0 a8
0020  00 3b f3 7a af 12 00 00 00 00 00 00 00 00 50 02
0030  20 00 05 73 00 00 66 00 00 00 00 00 00 00 00 00
0040  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

```

Figure 13. Hexadecimal View of Example TCP ENIP UnRegisterSession Request Packet

```

###[ ENIP UDP ]###
    Command  = 0x66
    Length   = None
    Session_Handle= 0x0
    Status    = Success
    Sender_Context= 0
    Options   = 0
###[ Unregister Session ]###

```

Fields encapsulated at the ENIP layer are highlighted.

Figure 14. An Example ENIP UnRegisterSession Request over UDP Packet.

```

0000  45 00 00 34 00 01 00 00 40 11 f8 ee c0 a8 00 3e
0010  c0 a8 00 3b f3 7a af 12 00 20 75 56 66 00 00 00
0020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0030  00 00 00 00

```

Figure 15. Hexadecimal View of Example ENIP UnRegisterSession Request over UDP Packet

3. ENIP SendRRData Command

SendRRData sends encapsulated messages from an originator to a target. When encapsulating CIP, the SendRRData command transports unconnected messages [16]. To test the SendRRData command fields, the following fields are fuzzed using a TCP connection: Session Handle, Status, Sender Context, Options, Interface Handle, and Timeout fields. The Session Handle field is tested with a combination of inputs ranging from 0x00000000 to 0xFFFFFFFF. The Status field is fuzzed in a similar manner. The Sender Context field is tested with data ranging from 0x0000000000000000 to 0xFFFFFFFFFFFFFFFF. The Options field is tested between 0x00000000 and

0xFFFFFFFF. The Interface Handle is tested between 0x00000000 and 0xFFFFFFFF and the Timeout field is tested between 0 and 65535. For the Encapsulated Data field, a CIP Forward Open command is used. Figures 16 and 17 illustrate a sample SendRRData request containing an encapsulated CIP Forward Open Request [15].


```

###[ ENIP TCP ]###
  Command  = Send RR Data (0x006F)
  Length   = 62
  Session_Handle= 0x0
  Status    = Success
  Sender_Context= 0
  Options   = 0
###[ Send RR Data ]###
  Interface_Handle= 0
  Timeout   = 0
###[ ENIP_CommonPacketFormat ]###
  Item_Count= None
  \Items
  \
  |###[ Common Packet Format Item ]###
  |  Address_Data_Item= Null (0x0000)
  |  Address_Length= None
  |###[ Common Packet Format Item ]###
  |  Address_Data_Item= Unconnected Message (0x00B2)
  |  Data_Length= 46
###[ Common_Industrial_Protocol ]###
  Request_Response= Request
  Common_Service= Connection_Manager_Forward_Open
  Request_Path_Size= None
  \Words
  \
  |###[ CIP Request Path ]###
  |  Path_Segment_Type= Logical Segment
  |  Logical_Segment_Type= Class ID
  |  Logical_Segment_Format= 8-bit logical address
  |  Class = Connection Manager Object
  |###[ CIP Request Path ]###
  |  Path_Segment_Type= Logical Segment
  |  Logical_Segment_Type= Instance ID
  |  Logical_Segment_Format= 8-bit logical address
  |  Eight_bit_Instance= 0x1
###[ CIP CM Forward Open Request ]###
  Reserved = 512
  Priority = Normal
  Tick_Time = 128
  Time_out_ticks= 155
  O_T_Network_Connection_ID= 0x80000002
  T_O_Network_Connection_ID= 0x80fe0001
  Connection_Serial_Number= 0x2
  Originator_Vendor_ID= Rockwell Software, Inc.
  Originator_Serial_Number= 90180339
  Connection_Timeout_Multiplier= 2
  Reserved = 0x200
  O_T_RPI = 1250000
  T_O_Network_Connection_Parameters= 0x4312
  T_O_RPI = 1250000
  T_O_Network_Connection_Parameters= 0x4312
  Transport_Type_Trigger_Direction= Server
  Transport_Type_Trigger_Production_Trigger=

Application Object
  Transport_Type_Trigger_Transport_Class= Class 3
  Connection_Path_Size= None
  \Words
  \
  |###[ CIP Request Path ]###
  |  Path_Segment_Type= Logical Segment
  |  Logical_Segment_Type= Class ID
  |  Logical_Segment_Format= 8-bit logical address
  |  Class = Message Router
  |###[ CIP Request Path ]###
  |  Path_Segment_Type= Logical Segment
  |  Logical_Segment_Type= Instance ID
  |  Logical_Segment_Format= 8-bit logical address
  |  Eight_bit_Instance= 0x1

```

Fields encapsulated at the ENIP layer are highlighted.

Figure 16. An Example ENIP SendRRData Request over TCP with an Encapsulated CIP Forward Open Request

0000	00	1d	9c	ca	cb	1b	90	e2	ba	18	fc	2e	08	00	45	00
0010	00	7e	00	01	00	00	40	06	f8	af	c0	a8	00	3e	c0	a8
0020	00	3b	f3	7a	af	12	00	00	00	00	00	00	00	00	50	02
0030	20	00	c2	cb	00	00	6f	00	3e	00	00	00	00	00	00	00
0040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0050	00	00	00	00	02	00	00	00	00	00	b2	00	2e	00	54	02
0060	20	06	24	01	07	9b	02	00	00	80	01	00	fe	80	02	00
0070	4d	00	f3	0a	60	05	02	00	02	00	d0	12	13	00	12	43
0080	d0	12	13	00	12	43	a3	02	20	02	24	01				

Figure 17. Hexadecimal View of Example ENIP SendRRData Request over TCP Packet

4. ENIP SendUnitData Command

The SendUnitData command [16] sends encapsulated connected messages that rely on their own end-to-end transport. Both originators and targets can initiate the SendUnitData command over a TCP connection. SendUnitData and SendRRData use the same packet structure. The Session Handle, Status, Sender Context, Options, Interface Handle, and Timeout fields are tested in the same manner as for SendRRData. Figures 18 and 19 demonstrate a sample SendRRData packet structure.

```

###[ ENIP TCP ]###
    Command    = Send Unit Data (0x0070)
    Length     = 28
    Session_Handle= 0x0
    Status     = Success
    Sender_Context= 0
    Options    = 0
###[ Send Unit Data ]###
    Interface_Handle= 0
    Timeout      = 0
###[ ENIP_CommonPacketFormat ]###
    Item_Count= None
    \Items      \
    |###[ Common Packet Format Item ]###
    | Address_Data_Item= Connection-Based (0x00A1)
    | Address_Length= 4
    | Connection_Identifier= 0x0
    |###[ Common Packet Format Item ]###
    | Address_Data_Item= Connected Transport Packet (0x00B1)
    | Data_Length= 8
    | Sequence_Number= 0x2
###[ Common_Industrial_Protocol ]###
    Request_Response= Request
    Common_Service= Get_Attributes_All
    Request_Path_Size= 2
    \Words          \
    |###[ CIP Request Path ]###
    | Path_Segment_Type= Logical Segment
    | Logical_Segment_Type= Class ID
    | Logical_Segment_Format= 8-bit logical address
    | Class      = Identity Object
    |###[ CIP Request Path ]###
    | Path_Segment_Type= Logical Segment
    | Logical_Segment_Type= Instance ID
    | Logical_Segment_Format= 8-bit logical address
    | Eight_bit_Instance= 0x1

```

Fields encapsulated at the ENIP layer are highlighted.

Figure 18. An Example ENIP SendUnitData Request over TCP with an Encapsulated CIP Get_Attribute_All Request

0000	00 1d 9c ca cb 1b 90 e2 ba 18 fc 2e 08 00 45 00
0010	00 5c 00 01 00 00 40 06 f8 ef c0 a8 00 3e c0 a8
0020	00 1d f3 7a af 12 00 00 00 00 00 00 00 00 50 02
0030	20 00 38 6f 00 00 70 00 1c 00 00 00 00 00 00 00
0040	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0050	00 00 00 00 02 00 a1 00 04 00 00 00 00 00 00 b1 00
0060	08 00 02 00 01 02 20 01 24 01

Figure 19. Hexadecimal View of Example EtherNet/IP SendUnitData Request over TCP Packet

5. ENIP Reserved for Legacy Use Commands

In the CIP Networks Library: Volume 2 EtherNet/IP Adaptation of CIP specification [16], several commands are labeled as “Reserved for legacy use” (herein referred to as Legacy Use) with no explanation of their functionality or packet structure. The command codes for the Legacy Use commands are 0x0001, 0x0002, 0x0005, 0x0067-0x006E, and 0x0071-0x00C7. These commands are tested to determine if MicroLogix handles them as defined by the ENIP specification, i.e., commands that are not supported by a target device shall not break the session or TCP connection. This testing also aims to discover unknown functionality of the legacy commands. Testing is conducted over both TCP and UDP connections. Figures 20 and 21 show the structure of a sample ENIP Legacy Use command sent over TCP. Figures 22 and 23 show the structure of a sample Legacy Use command sent over UDP.

```

###[ ENIP TCP ]###
  Command   = Indicate Status (0x0072)
  Length    = None
  Session_Handle = 0x0
  Status    = Success
  Sender_Context = 0
  Options   = 0

```

Fields encapsulated at the ENIP layer are highlighted.

Figure 20. An Example ENIP Legacy Use Request over TCP

```

0000  00 1d 9c ca cb 1b 90 e2 ba 18 fc 2e 08 00 45 00
0010  00 40 00 01 00 00 40 06 f8 ed c0 a8 00 3e c0 a8
0020  00 3b f3 7a af 12 00 00 00 00 00 00 00 00 50 02
0030  20 00 f9 72 00 00 72 00 00 00 00 00 00 00 00 00
0040  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

```

Figure 21. Hexadecimal View of Example ENIP Legacy Use Request over TCP Packet


```

###[ ENIP UDP ]###
Command    = 0x72
Length     = None
Session_Handle= 0x0
Status     = Success
Sender_Context= 0
Options    = 0

```

Fields encapsulated at the ENIP layer are highlighted.

Figure 22. An Example ENIP Legacy Use Request over UDP.

```

0000  45 00 00 34 00 01 00 00 40 11 f8 ee c0 a8 00 3e
0010  c0 a8 00 3b f3 7a af 12 00 20 69 56 72 00 00 00
0020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0030  00 00 00 00

```

Figure 23. Hexadecimal View of Example ENIP Legacy Use Request over UDP Packet

6. ENIP Reserved for Future Use Commands

There are also designated ENIP commands that are labeled “Reserved for future use” (herein referred to as Future Use) in the CIP Networks Library: Volume 2 EtherNet/IP Adaptation of CIP specification [16]. The ranges of the Future Use commands are 0x0006-0x0062 and 0x00C8-0xFFFF. These commands are tested to determine if MicroLogix handles them as defined by the ENIP specification, i.e., commands that are not supported by a target device shall not break the session or TCP connection. Figures 24 and 25 show the structure of an ENIP Future Use command sent over TCP. Figures 26 and 27 show the structure of an ENIP Future Use command sent over UDP.

```

###[ ENIP TCP ]###
Command   = Unknown Command (0x0006)
Length    = None
Session_Handle= 0x0
Status    = Success
Sender_Context= 0
Options   = 0

```

Fields encapsulated at the ENIP layer are highlighted.

Figure 24. An Example ENIP Future Use Request over TCP

```

0000  00 1d 9c ca cb 1b 90 e2 ba 18 fc 2e 08 00 45 00
0010  00 40 00 01 00 00 40 06 f8 ed c0 a8 00 3e c0 a8
0020  00 3b f3 7a af 12 00 00 00 00 00 00 00 00 00 02
0030  20 00 f9 72 00 00 06 00 00 00 00 00 00 00 00 00
0040  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

```

Figure 25. Hexadecimal View of Example ENIP Future Use Request over TCP Packet

```

###[ ENIP UDP ]###
Command   = 0x06
Length    = None
Session_Handle= 0x0
Status    = Success
Sender_Context= 0
Options   = 0

```

Fields encapsulated at the ENIP layer are highlighted.

Figure 26. An Example UDP ENIP Future Use Request

```

0000  45 00 00 34 00 01 00 00 40 11 f8 ee c0 a8 00 3e
0010  c0 a8 00 3b f3 7a af 12 00 20 69 56 06 00 00 00
0020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0030  00 00 00 00

```

Figure 27. Hexadecimal View of Example UPD ENIP Future Use Response Packet

C. CIP FUZZING TEST PLAN

Previous CIP fuzz testing is limited to the CIP NOP command [9]. This thesis expands the testing scope to include four additional CIP Common Services shown in the CIP Test Plan in Table 7. Command fields are tested in isolation. All tests use the ENIP command SendUnitData, which can only be used with TCP.

Table 7. CIP Test Plan

Test Number	CIP Command	Fuzzed Field	Protocol	Fuzzing Parameters
T30	Get_Attributes_All	Class	TCP	Class 0x00-0xFF, Attribute 0x01
T31	Get_Attributes_All	Instance	TCP	Class 0x01, Attribute 0x00-0xFF
T32	Get_Attribute_List	Class	TCP	Class 0x00-0xFF, Attribute_List 0x01, Instance 0x01
T33	Get_Attribute_List	Attribute_List	TCP	Class 0x01, Attribute_List 0x00-0xFF, Instance 0x01
T34	Get_Attribute_List	Instance	TCP	Class 0x01, Attribute_List 0x01, Instance 0x00-0xFF
T35	Get_Attribute_List	Attribute_count	TCP	Max Attribute_count Length (Increasing lengths of Attribute_count field)
T36	Get_Attribute_Single	Class	TCP	Class 0x00-0xFF, Attribute 0x01, Instance 0x00
T37	Get_Attribute_Single	Instance	TCP	Class 0x01, Attribute 0x01, Instance 0x00-0xFF
T38	Get_Attribute_Single	Attribute	TCP	Class 0x01, Attribute 0x00-0xFF, Instance 0x01
T39	Find_Next_Object_Instance	Class	TCP	Class 0x00-0xFF, Instance 0x00, Maximum Returned Values 0x00
T40	Find_Next_Object_Instance	Instance	TCP	Class 0x01, Instance 0x00-0xFF, Maximum Returned Values 0x00
T41	Find_Next_Object_Instance	Maximum Returned Values	TCP	Class 0x01, Instance 0x00, Maximum Returned Values 0x00

1. CIP Get_Attributes_All

The Get_Attributes_All command requests the contents of all instance or class attributes that the specified object supports [15]. Both Class and Attribute fields are individually fuzzed with values in the range of 0x00 to 0xFF. Figures 28 and 29 show the structure of a sample Get_Attributes_All command over TCP.

```

###[ ENIP TCP ]###
  Command   = Send Unit Data (0x0070)
  Length    = 28
  Session_Handle= 0x0
  Status    = Success
  Sender_Context= 0
  Options   = 0
###[ Send Unit Data ]###
  Interface_Handle= 0
  Timeout       = 0
###[ ENIP_CommonPacketFormat ]###
  Item_Count= None
  \Items
  \
  |###[ Common Packet Format Item ]###
  |  Address_Data_Item= Connection-Based (0x00A1)
  |  Address_Length= 4
  |  Connection_Identifier= 0x0
  |###[ Common Packet Format Item ]###
  |  Address_Data_Item= Connected Transport Packet (0x00B1)
  |  Data_Length= 8
  |  Sequence_Number= 0x2
###[ Common_Industrial_Protocol ]###
  Request_Response= Request
  Common_Service= Get Attributes All
  Request_Path_Size= 2
  \Words
  \
  |###[ CIP Request Path ]###
  |  Path_Segment_Type= Logical Segment
  |  Logical_Segment_Type= Class ID
  |  Logical_Segment_Format= 8-bit logical address
  |  Class = Identity Object
  |###[ CIP Request Path ]###
  |  Path_Segment_Type= Logical Segment
  |  Logical_Segment_Type= Instance ID
  |  Logical_Segment_Format= 8-bit logical address
  |  Eight_bit_Instance= 0x1

```

Fields encapsulated at the CIP layer are highlighted.

Figure 28. An Example CIP Get_Attributes_All Request over TCP

0000	00 1d 9c ca cb 1b 90 e2 ba 18 fc 2e 08 00 45 00
0010	00 5c 00 01 00 00 40 06 f8 ef c0 a8 00 3e c0 a8
0020	00 1d f3 7a af 12 00 00 00 00 00 00 00 00 50 02
0030	20 00 38 6f 00 00 70 00 1c 00 00 00 00 00 00 00
0040	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0050	00 00 00 00 02 00 a1 00 04 00 00 00 00 00 b1 00
0060	08 00 02 00 01 02 20 01 24 01

Figure 29. Hexadecimal View of Example CIP Get_Attributes_All Request over TCP Packet

2. CIP Get_Attribute_List

The Get_Attribute_List Command requests the selected attributes of an object class or instance [15]. The Get_Attribute_List is an optional service [2]. The Class, Attribute, and Instance fields are individually fuzzed with values in the range of 0x00-0xFF. The Attribute_count field is also tested by sending Get_Attribute_List requests with increasing values in the Attribute_count field up to 0xFFFF to determine the maximum number of attributes allowable. Figures 30 and 31 show the structure of a sample TCP Get_Attribute_List command.

```

###[ ENIP TCP ]###
    Command    = Send Unit Data (0x0070)
    Length     = 32
    Session_Handle= 0xf020100
    Status     = Success
    Sender_Context= 0
    Options    = 0
###[ Send Unit Data ]###
    Interface_Handle= 0
    Timeout     = 1
###[ ENIP_CommonPacketFormat ]###
    Item_Count= 2
    \Items      \
        |###[ Common Packet Format Item ]###
        | Address_Data_Item= Connection-Based (0x00A1)
        | Address_Length= 4
        | Connection_Identifier= 0x9f9d0b6f
        |###[ Common Packet Format Item ]###
        | Address_Data_Item= Connected Transport Packet (0x00B1)
        | Data_Length= 14
        | Sequence_Number= 0x1
###[ Common_Industrial_Protocol ]###
    Request_Response= Request
    Common_Service= Get Attribute List
    Request_Path_Size= 2
    \Words      \
        |###[ CIP Request Path ]###
        | Path_Segment_Type= Logical Segment
        | Logical_Segment_Type= Class ID
        | Logical_Segment_Format= 8-bit logical address
        | Class = Identity
        |###[ CIP Request Path ]###
        | Path_Segment_Type= Logical Segment
        | Logical_Segment_Type= Instance ID
        | Logical_Segment_Format= 8-bit logical address
        | Eight_bit_Instance= 0x1
###[ CIP Get Attribute List Request ]###
    Attribute_Count= 1
    Attributes= ['1']

```

Fields encapsulated at the CIP layer are highlighted.

Figure 30. An Example CIP Get_Attribute_List Request over TCP

0000	00	1d	9c	ca	cb	1b	90	e2	ba	18	fc	2e	08	00	45	00
0010	00	60	00	01	00	00	40	06	e4	92	0a	01	1e	01	0a	01
0020	64	02	f3	7a	af	12	00	00	00	00	00	00	00	00	50	02
0030	20	00	06	62	00	00	70	00	20	00	00	01	02	0f	00	00
0040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0050	00	00	01	00	02	00	a1	00	04	00	6f	0b	9d	9f	b1	00
0060	0e	00	01	00	03	02	20	01	24	01	01	00	01	00	01	00

Figure 31. Hexadecimal View of Example CIP Get_Attribute_List Request over TCP Packet

3. CIP Get_Attribute_Single

The Get_Attribute_Single command requests the contents of a specified attribute. This service is to be implemented for the Identity Object if any Class Attributes are implemented [15]. Class, Attribute, and Instance fields are fuzzed with values ranging from 0x00 to 0xFF. Figures 32 and 33 show the structure of a sample TCP Get_Attribute_Single command.

```

###[ ENIP TCP ]###
    Command = Send Unit Data (0x0070)
    Length = 30
    Session_Handle= 0xf020100
    Status = Success
    Sender_Context= 0
    Options = 0
###[ Send Unit Data ]###
    Interface_Handle= 0
    Timeout = 1
###[ ENIP_CommonPacketFormat ]###
    Item_Count= 2
    \Items
    \
    ###[ Common Packet Format Item ]###
    Address_Data_Item= Connection-Based (0x00A1)
    Address_Length= 4
    Connection_Identifier= 0x9f9d0b6f
    ###[ Common Packet Format Item ]###
    Address_Data_Item= Connected Transport Packet (0x00B1)
    Data_Length= 10
    Sequence_Number= 0x1
###[ Common_Industrial_Protocol ]###
    Request_Response= Request
    Common_Service= Get Attribute Single
    Request_Path_Size= 2
    \Words
    \
    ###[ CIP Request Path ]###
    Path_Segment_Type= Logical Segment
    Logical_Segment_Type= Class ID
    Logical_Segment_Format= 8-bit logical address
    Class = Identity Object
    ###[ CIP Request Path ]###
    Path_Segment_Type= Logical Segment
    Logical_Segment_Type= Instance ID
    Logical_Segment_Format= 8-bit logical address
    Eight_bit_Instance= 0x1
###[ CIP Get Attribute Single Request ]###
    Attribute_Identifier= 1

```

Fields encapsulated at the CIP layer are highlighted.

Figure 32. An Example CIP Get_Attribute_Single Request over TCP

0000	00	1d	9c	ca	cb	1b	90	e2	ba	18	fc	2e	08	00	45	00
0010	00	5d	00	01	00	00	40	06	e4	95	0a	01	1e	01	0a	01
0020	64	02	f3	7a	af	12	00	00	00	00	00	00	00	00	50	02
0030	20	00	05	5a	00	00	70	00	1d	00	00	01	02	0f	00	00
0040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0050	00	00	01	00	02	00	a1	00	04	00	6f	0b	9d	9f	b1	00
0060	09	00	01	00	0e	02	20	01	24	01	01	00				

Figure 33. Hexadecimal View of Example CIP Get_Attribute_Single Request over TCP Packet

4. CIP Find_Next_Object_Instance

The Find_Next_Object_Instance command requests a list of Instance IDs associated with existing Object Instances that are accessible from the CIP subnet at the time the request is made [15]. The request command specifies the number of requested Instances, but the number of returned Instances can be less. If the Instance ID in the request is zero, the Instance ID that is numerically lowest in the Class is returned [15]. If the Instance ID in the request is less than the highest Instance ID in the Class, successful responses return the next Instance ID that is numerically higher than the Instance ID specified in the request [2]. If the Instance ID in the request is greater than or equal to the highest Instance ID in the Class, the value 0 is returned [15]. This service is only available at the Class level [15]. Testing is conducted on the Class, Instance, and Maximum Returned Values fields of this command with inputs ranging from 0x00 to 0xFF. Figures 34 and 35 show the structure of a sample CIP Find_Next_Object_Instance command over TCP.

```

###[ ENIP TCP ]###
    Command = Send RR Data (0x006F)
    Length = None
    Session_Handle= 0xf020100
    Status = Success
    Sender_Context= 0
    Options = 0
###[ Send RR Data ]###
    Interface_Handle= 0
    Timeout = 0
###[ ENIP_CommonPacketFormat ]###
    Item_Count= None
    \Items \
        ###[ Common Packet Format Item ]###
        Address_Data_Item= Null (0x0000)
        Address_Length= 4
        ###[ Common Packet Format Item ]###
        Address_Data_Item= Unconnected Message (0x00B2)
        Data_Length= 7
###[ Common_Industrial_Protocol ]###
    Request_Response= Request
    Common_Service= Find Next Object Instance
    Request_Path_Size= 2
    \Words \
        ###[ CIP Request Path 1 ]###
        Path_Segment_Type= Logical Segment
        Logical_Segment_Type= Class ID
        Logical_Segment_Format= 8-bit logical address
        Class = Identity Object
        ###[ CIP Request Path 1 ]###
        Path_Segment_Type= Logical Segment
        Logical_Segment_Type= Instance ID
        Logical_Segment_Format= 8-bit logical address
        Eight_bit_Instance= 0x1
###[ CIP_Find_Next_Object_Instance_Request ]###
    Max_Returned_Vals= 1

```

Fields encapsulated at the CIP layer are highlighted.

Figure 34. An Example CIP Find_Next_Object_Instance Request over TCP

0000	00 1d 9c ca cb 1b 90 e2 ba 18 fc 2e 08 00 45 00
0010	00 57 00 01 00 00 40 06 f8 d6 c0 a8 00 3e c0 a8
0020	00 3b f3 7a af 12 00 00 00 00 00 00 00 00 50 02
0030	20 00 ce 46 00 00 6f 00 17 00 00 01 02 0f 00 00
0040	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0050	00 00 00 00 02 00 00 00 04 00 b2 00 07 00 11 02
0060	20 01 24 01 01

Figure 35. Hexadecimal View of Example CIP Find_Next_Object_Instance Request over TCP Packet

D. PCCC FUZZING TEST PLAN

Previous MicroLogix PCCC fuzz testing is limited to the Execute PCCC command Protected Typed Logical Read with Three Address Fields [17]. This thesis expands the MicroLogix testing to fuzz PCCC commands not tested by Tacliad [9] for vulnerabilities. Command fields are tested in isolation on the MicroLogix PLC in order to provide a methodical evaluation of each command's potential vulnerabilities.

Additionally, to determine if a recently discovered MicroLogix PCCC vulnerability affects the ControlLogix, the Protected Logical Read with Three Address Fields is tested on the ControlLogix with a MicroLogix fault-causing combination of field inputs. Table 8 summarizes the PCCC test plan.

Table 8. PCCC Testing Plan

Test Number	PCCC Command	Fuzzed Field	Protocol	Fuzzing Parameters
MicroLogix Tests				
T42	Echo	Data: 0 bytes	TCP	0 Attached bytes
T43	Echo	Data: Max Length	TCP	Increasing number of attached bytes
T44	Echo	Data: 8 bytes	TCP	8 Attached random bytes
T45	Echo	Data: 9 bytes	TCP	9 Attached random bytes
T46	Echo	Data: 10 bytes	TCP	10 Attached random bytes
T47	Echo	Data: 40 bytes	TCP	40 Attached random bytes
T48	Echo	Data: 243 bytes	TCP	243 Attached random bytes
T49	Echo	Data: Maximum bytes returned by module with no errors	TCP	Maximum random bytes returned by module with no error
T50	Echo	Data: 248 bytes	TCP	248 Attached random bytes
T51	Echo	Data: 256 bytes	TCP	256 Attached random bytes
T52	Protected Typed File Read	Size	TCP	Size (0x00-0xFF)

Test Number	PCCC Command	Fuzzed Field	Protocol	Fuzzing Parameters
T53	Protected Typed File Read	Tag	TCP	Tag (0x0000-0xFFFF)
T54	Protected Typed File Read	Offset	TCP	Offset (0x0000-0xFFFF)
T55	Protected Typed File Read	File Type	TCP	File Type (0x00-0xFF)
T56	Protected Typed File Write	Size	TCP	Size (0x00-0xFF)
T57	Protected Typed File Write	Tag	TCP	Tag (0x0000-0xFFFF)
T58	Protected Typed File Write	Offset	TCP	Offset (0x0000-0xFFFF)
T59	Protected Typed File Write	File Type	TCP	File Type (0x00-0xFF)
T60	Protected Typed File Write	Data	TCP	Data (0x00-0xFF)
T61	Protected Typed Logical Write with Three Address Fields	Byte Size	TCP	Byte Size (0x00-0xFF)
T62	Protected Typed Logical Write with Three Address Fields	File No.	TCP	File No. (0x00-0xFF)
T63	Protected Typed Logical Write with Three Address Fields	File Type	TCP	File Type (0x00-0xFF)
T64	Protected Typed Logical Write with Three Address Fields	Element No.	TCP	Element No. (0x00-0xFF and 0xFF0000-0xFFFFFFFF)
T65	Protected Typed Logical Write with Three Address Fields	Sub-Element No.	TCP	Sub-Element No. (0x00-0xFF and 0xFF0000-0xFFFFFFFF)
T66	Unprotected Read	Address	TCP	Address (0x0000-0xFFFF)
T67	Unprotected Read	Size	TCP	Size (0x00-0xFF)
T68	Diagnostic Status-Functionality Test	N/A	TCP	Properly formatted command
T69	Read Diagnostic Counters	Address	TCP	Address (0x0000-0xFFFF)

Test Number	PCCC Command	Fuzzed Field	Protocol	Fuzzing Parameters
T70	Read Diagnostic Counters	Size	TCP	Size (0x00-0xFF)
T71	Restart-Functionality Test	N/A	TCP	Properly formatted command
T72	Download Completed-Functionality Test	N/A	TCP	Properly formatted command
ControlLogix Tests				
T73	Protected Typed Logical Read with Three Address Fields	File No., File Type	TCP	File No. (0x2-0x8), File Type (0x47-0x48)

1. PCCC Echo Command

The Echo command enables a user to check the integrity of a communication link. The receiving module replies to a request with the same data in the original transmission. According Allen-Bradley's DF1 Protocol and Command Set specification [17], this command is compatible with the MicroLogix 1000, a member of the MicroLogix 1100 family of products, and should transmit a maximum of 243 bytes of data. In order to test the maximum data allowable in an Echo command, the fuzzing device sends commands with an increasing number of repeating bytes, starting from 0 to the maximum size that the receiving module will reply with no errors, while monitoring SUT responses. Echo commands are tested with random bytes using the following lengths: 0, 8, 9, 10, 243, 248, 256, and the observed maximum size returned with no errors. Figures 36 and 37 illustrate the structure of the PCCC Echo command.


```

###[ ENIP TCP ]###
    Command    = Send Unit Data (0x0070)
    Length     = 42
    Session_Handle= 0xf020100
    Status      = Success
    Sender_Context= 0
    Options     = 0
###[ Send Unit Data ]###
    Interface_Handle= 0
    Timeout      = 1
###[ ENIP_CommonPacketFormat ]###
    Item_Count= 2
    \Items      \
        ###[ Common Packet Format Item ]###
        Address_Data_Item= Connection-Based (0x00A1)
        Address_Length= 4
        Connection_Identifier= 0x9f9d0b6f
        ###[ Common Packet Format Item ]###
        Address_Data_Item= Connected Transport Packet (0x00B1)
        Data_Length= 23
        Sequence_Number= 0x1
###[ Common_Industrial_Protocol ]###
    Request_Response= Request
    Common_Service= Execute_PCCC_Service
    Request_Path_Size= 2
    \Words      \
        ###[ CIP Request Path ]###
        Path_Segment_Type= Logical Segment
        Logical_Segment_Type= Class ID
        Logical_Segment_Format= 8-bit logical address
        Class      = 0x67
        ###[ CIP Request Path ]###
        Path_Segment_Type= Logical Segment
        Logical_Segment_Type= Instance ID
        Logical_Segment_Format= 8-bit logical address
        Eight_bit_Instance= 0x1
###[ CIP Execute PCCC Service Request ]###
    Length_of_Requestor_ID= 7
    CIP_Vendor_ID_of_Requestor= Rockwell Software, Inc.
    CIP_Serial_Number= 90180339
    CMD      = 0x06
    Status    = 0x0
    Transaction_Word= 1
    Function  = Echo
###[ Padding ]###
    load      = '\x01\x02'

```

Fields encapsulated at the PCCC layer are highlighted.

Figure 36. An Example PCCC Echo Request with Two Data Bytes over TCP

0000	00	1d	9c	ca	cb	1b	90	e2	ba	18	fc	2e	08	00	45	00
0010	00	6a	00	01	00	00	40	06	e4	88	0a	01	1e	01	0a	01
0020	64	02	f3	7a	af	12	00	00	00	00	00	00	00	00	50	02
0030	20	00	95	3d	00	00	70	00	2a	00	00	01	02	0f	00	00
0040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0050	00	00	01	00	02	00	a1	00	04	00	6f	0b	9d	9f	b1	00
0060	17	00	01	00	4b	02	20	67	24	01	07	4d	00	f3	0a	60
0070	05	06	00	01	00	00	01	02								

Figure 37. Hexadecimal View of Example PCCC Echo Request over TCP Packet.

2. PCCC Protected Typed File Read

The Protected Typed File Read command reads data from an open file [17]. Four fields are fuzz tested: Size, Tag, Offset, and File Type. The one-byte fields, Size and File Type, are tested with random inputs from 0x00 to 0xFF. The two-byte fields, Tag and Offset, are tested with random inputs from 0x0000 to 0xFFFF. The SUT is expected to provide successful read responses. Figures 38 and 39 illustrate the structure of an example packet.

```

###[ ENIP TCP ]###
  Command      = Send Unit Data (0x0070)
  Length       = None
  Session_Handle= 0xf020100
  Status       = Success
  Sender_Context= 0
  Options      = 0
###[ Send Unit Data ]###
  Interface_Handle= 0
  Timeout       = 1
###[ ENIP_CommonPacketFormat ]###
  Item_Count= None
  \Items
  |###[ Common Packet Format Item ]###
  |  Address_Data_Item= Connection-Based (0x00A1)
  |  Address_Length= 4
  |  Connection_Identifier= 0x9f9d0b6f
  |###[ Common Packet Format Item ]###
  |  Address_Data_Item= Connected Transport Packet (0x00B1)
  |  Data_Length= 26
  |  Sequence_Number= 0x1
###[ Common_Industrial_Protocol ]###
  Request_Response= Request
  Common_Service= Execute_PCCC_Service
  Request_Path_Size= None
  \Words
  |###[ CIP Request Path ]###
  |  Path_Segment_Type= Logical Segment
  |  Logical_Segment_Type= Class ID
  |  Logical_Segment_Format= 8-bit logical address
  |  Class = 0x67
  |###[ CIP Request Path ]###
  |  Path_Segment_Type= Logical Segment
  |  Logical_Segment_Type= Instance ID
  |  Logical_Segment_Format= 8-bit logical address
  |  Eight_bit_Instance= 0x1
###[ CIP Execute PCCC Service Request ]###
  Length_of_Requestor_ID= 7
  CIP_Vendor_ID_of_Requestor= Rockwell Software, Inc.
  CIP_Serial_Number= 90180339
  CMD = 0x0F
  Status = 0x0
  Transaction_Word= 2
  Function = Protected Typed File Read
  Size = 0xc0
  Tag = 33765
  Offset = 36443
  File_Type = 0xc0

```

Fields encapsulated at the PCCC layer are highlighted.

Figure 38. An Example PCCC Protected Typed File Read Request

0000	00	1d	9c	ca	cb	1b	90	e2	ba	18	fc	2e	08	00	45	00
0010	00	6e	00	01	00	00	40	06	f8	bf	c0	a8	00	3e	c0	a8
0020	00	3b	f3	7a	af	12	00	00	00	00	00	00	00	00	50	02
0030	20	00	cf	c3	00	00	70	00	2e	00	00	01	02	0f	00	00
0040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0050	00	00	01	00	02	00	a1	00	04	00	6f	0b	9d	9f	b1	00
0060	1a	00	01	00	4b	02	20	67	24	01	07	4d	00	f3	0a	60
0070	05	0f	00	02	00	a7	c0	e5	83	5b	8e	c0				

Figure 39. Hexadecimal View of Example PCCC Protected Typed File Read Packet

3. PCCC Protected Typed File Write

The Protected Typed File Write command writes data to an open file in the PLC [17]. Testing is conducted on five fields: Size, Tag, Offset, File Type, and Data. The one-byte fields, Size and File Type, are tested with random inputs from 0x00 to 0xFF. The two-byte fields, Tag and Offset, are tested with random inputs from 0x0000 to 0xFFFF. The data field is tested with a two-byte size with random inputs from 0x0000 to 0xFFFF. The SUT is expected to provide successful write responses. Figures 40 and 41 illustrate the structure of an example packet.

```

###[ ENIP TCP ]###
  Command = Send Unit Data (0x0070)
  Length = None
  Session_Handle= 0xf020100
  Status = Success
  Sender_Context= 0
  Options = 0
###[ Send Unit Data ]###
  Interface_Handle= 0
  Timeout = 1
###[ ENIP_CommonPacketFormat ]###
  Item_Count= None
  \Items
  |###[ Common Packet Format Item ]###
  |  Address_Data_Item= Connection-Based (0x00A1)
  |  Address_Length= 4
  |  Connection_Identifier= 0x9f9d0b6f
  |###[ Common Packet Format Item ]###
  |  Address_Data_Item= Connected Transport Packet (0x00B1)
  |  Data_Length= 28
  |  Sequence_Number= 0x1
###[ Common_Industrial_Protocol ]###
  Request_Response= Request
  Common_Service= Execute_PCCC_Service
  Request_Path_Size= None
  \Words
  |###[ CIP Request Path ]###
  |  Path_Segment_Type= Logical Segment
  |  Logical_Segment_Type= Class ID
  |  Logical_Segment_Format= 8-bit logical address
  |  Class = 0x67
  |###[ CIP Request Path ]###
  |  Path_Segment_Type= Logical Segment
  |  Logical_Segment_Type= Instance ID
  |  Logical_Segment_Format= 8-bit logical address
  |  Eight_bit_Instance= 0x1
###[ CIP Execute PCCC Service Request ]###
  Length_of_Requestor_ID= 7
  CIP_Vendor_ID_of_Requestor= Rockwell Software, Inc.
  CIP_Serial_Number= 90180339
  CMD = 0x0F
  Status = 0x0
  Transaction_Word= 2
  Function = Protected_Typed_File_Write
  Size = 0x2
  Tag = 4176
  Offset = 0
  File_Type = 0x5a
  Data = 65535

```

Fields encapsulated at the PCCC layer are highlighted.

Figure 40. An Example PCCC Protected Typed File Write Request

0000	00	1d	9c	ca	cb	1b	90	e2	ba	18	fc	2e	08	00	45	00
0010	00	70	00	01	00	00	40	06	f8	bd	c0	a8	00	3e	c0	a8
0020	00	3b	f3	7a	af	12	00	00	00	00	00	00	00	00	50	02
0030	20	00	8c	11	00	00	70	00	30	00	00	01	02	0f	00	00
0040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0050	00	00	01	00	02	00	a1	00	04	00	6f	0b	9d	9f	b1	00
0060	1c	00	01	00	4b	02	20	67	24	01	07	4d	00	f3	0a	60
0070	05	0f	00	02	00	af	02	50	10	00	00	5a	ff	ff		

Figure 41. Hexadecimal View of Example PCCC Protected Typed File Write Packet

4. PCCC Protected Typed Logical Write with Three Address Fields

The Protected Logical Write with Three Address Fields command writes data to a logical address in the PLC's processor [17]. The specification [17] is unclear whether the MicroLogix family of PLCs supports this command. Specifically, while the table that summarizes the PCCC commands and compatible processors indicates MicroLogix supports the command, the detailed description of this particular command omits MicroLogix as a supporting platform. Based on previous testing of the Protected Logical Read with Three Address Fields command [9], the MicroLogix is assumed to support the command. Testing is conducted on the fields Byte Size, File Number, and File Type with inputs ranging from 0x00 to 0xFF. Element Number, and Sub-element Number are one-byte fields that can expand to three bytes when the first byte is set to 0xFF. In this case, the second and third bytes identify the expanded sub-element [17]. For this reason, these fields are tested in the one-byte configuration with inputs ranging from 0x00 to 0xFF and in the three-byte configuration with inputs ranging from 0xFF0000 to 0xFFFFFFFF. The Data field is not fuzzed in an effort to avoid overwriting memory space with unknown functionality. Figures 42 and 43 show the structure of an example packet.

```

###[ ENIP TCP ]###
    Command    = Send Unit Data (0x0070)
    Length     = 45
    Session_Handle= 0xf020100
    Status      = Success
    Sender_Context= 0
    Options     = 0
###[ Send Unit Data ]###
    Interface_Handle= 0
    Timeout      = 1
###[ ENIP_CommonPacketFormat ]###
    Item_Count= None
    \Items      \
    |###[ Common Packet Format Item ]###
    | Address_Data_Item= Connection-Based (0x00A1)
    | Address_Length= 4
    | Connection_Identifier= 0x9f9d0b6f
    |###[ Common Packet Format Item ]###
    | Address_Data_Item= Connected Transport Packet (0x00B1)
    | Data_Length= 25
    | Sequence_Number= 0x1
###[ Common_Industrial_Protocol ]###
    Request_Response= Request
    Common_Service= Execute_PCCC_Service
    Request_Path_Size= None
    \Words      \
    |###[ CIP Request Path ]###
    | Path_Segment_Type= Logical Segment
    | Logical_Segment_Type= Class ID
    | Logical_Segment_Format= 8-bit logical address
    | Class      = 0x67
    |###[ CIP Request Path ]###
    | Path_Segment_Type= Logical Segment
    | Logical_Segment_Type= Instance ID
    | Logical_Segment_Format= 8-bit logical address
    | Eight_bit_Instance= 0x1
###[ CIP Execute PCCC Service Request ]###
    Length_of_Requestor_ID= 7
    CIP_Vendor_ID_of_Requestor= Rockwell Software, Inc.
    CIP_Serial_Number= 90180339
    CMD      = 0x0F
    Status    = 0x0
    Transaction_Word= 2
    Function  = Protected_Typed_Logical_Write_Three_Address_Fields
    Byte_Size = 0x1
    File_No   = 0x1
    File_Type = 0x1
    Element_No= 0x0
    Sub_Element_No= 0x0
    Data      = None

```

Fields encapsulated at the PCCC layer are highlighted.

Figure 42. An Example PCCC Protected Typed Logical Write with Three Address Fields Request

0000	00	1d	9c	ca	cb	1b	90	e2	ba	18	fc	2e	08	00	45	00
0010	00	6e	00	01	00	00	40	06	f8	bf	c0	a8	00	3e	c0	a8
0020	00	3b	f3	7a	af	12	00	00	00	00	00	00	00	00	50	02
0030	20	00	a1	c1	00	00	70	00	2e	00	00	01	02	0f	00	00
0040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0050	00	00	01	00	02	00	a1	00	04	00	6f	0b	9d	9f	b1	00
0060	19	00	01	00	4b	02	20	67	24	01	07	4d	00	f3	0a	60
0070	05	0f	00	02	00	aa	01	01	01	00	00					

Figure 43. Hexadecimal View of Example PCCC Protected Typed Logical Write with Three Address Fields Request over TCP Packet

5. PCCC Unprotected Read

The Unprotected Read command requests data from a common interface file on the PLC [17] Fuzz testing is conducted on two fields: Address and Size. The two-byte Address field is fuzzed with random numbers between 0x0000 to 0xFFFF. The one-byte Size field is fuzzed with inputs between 0x00 to 0xFF. The expected result of the MicroLogix testing is a successful read response from the SUT. Figures 44 and 45 show the structure of an example packet.

```

#### ENIP TCP ####
    Command = Send Unit Data (0x0070)
    Length = None
    Session_Handle= 0xf020100
    Status = Success
    Sender_Context= 0
    Options = 0
#### Send Unit Data ####
    Interface_Handle= 0
    Timeout = 1
#### ENIP_CommonPacketFormat ####
    Item_Count= None
    \Items
    \
    #### Common Packet Format Item ####
    | Address_Data_Item= Connection-Based (0x00A1)
    | Address_Length= 4
    | Connection_Identifier= 0x9f9d0b6f
    | #### Common Packet Format Item ####
    | Address_Data_Item= Connected Transport Packet (0x00B1)
    | Data_Length= 22
    | Sequence_Number= 0x1
#### Common_Industrial_Protocol ####
    Request_Response= Request
    Common_Service= Execute_PCCC_Service
    Request_Path_Size= None
    \Words
    \
    #### CIP Request Path ####
    | Path_Segment_Type= Logical Segment
    | Logical_Segment_Type= Class ID
    | Logical_Segment_Format= 8-bit logical address
    | Class = 0x67
    | #### CIP Request Path ####
    | Path_Segment_Type= Logical Segment
    | Logical_Segment_Type= Instance ID
    | Logical_Segment_Format= 8-bit logical address
    | Eight_bit_Instance= 0x1
#### CIP Execute PCCC Service Request ####
    Length_of_Requestor_ID= 7
    CIP_Vendor_ID_of_Requestor= Rockwell Software, Inc.
    CIP_Serial_Number= 90180339
    CMD = Unprotected_Read
    Status = 0x0
    Transaction_Word= 2
    Address = 0
    Size = 0x1

```

Fields encapsulated at the PCCC layer are highlighted.

Figure 44. An Example PCCC Unprotected Read Request

```

#### ENIP TCP ####
    Command = Send Unit Data (0x0070)
    Length = None
    Session_Handle= 0xf020100
    Status = Success
    Sender_Context= 0
    Options = 0
#### Send Unit Data ####
    Interface_Handle= 0
    Timeout = 1
#### ENIP_CommonPacketFormat ####
    Item_Count= None
    \Items
    \
    #### Common Packet Format Item ####
    | Address_Data_Item= Connection-Based (0x00A1)
    | Address_Length= 4
    | Connection_Identifier= 0x9f9d0b6f
    | #### Common Packet Format Item ####
    | Address_Data_Item= Connected Transport Packet (0x00B1)
    | Data_Length= 22
    | Sequence_Number= 0x1
#### Common_Industrial_Protocol ####
    Request_Response= Request
    Common_Service= Execute_PCCC_Service
    Request_Path_Size= None
    \Words
    \
    #### CIP Request Path ####
    | Path_Segment_Type= Logical Segment
    | Logical_Segment_Type= Class ID
    | Logical_Segment_Format= 8-bit logical address
    | Class = 0x67
    | #### CIP Request Path ####
    | Path_Segment_Type= Logical Segment
    | Logical_Segment_Type= Instance ID
    | Logical_Segment_Format= 8-bit logical address
    | Eight_bit_Instance= 0x1
#### CIP Execute PCCC Service Request ####
    Length_of_Requestor_ID= 7
    CIP_Vendor_ID_of_Requestor= Rockwell Software, Inc.
    CIP_Serial_Number= 90180339
    CMD = Unprotected_Read
    Status = 0x0
    Transaction_Word= 2
    Address = 0
    Size = 0x1

```

Figure 45. Hexadecimal View of Example PCCC Unprotected Read Packet

6. PCCC Diagnostic Status

The Diagnostic Status command requests up to 244 bytes of status information from an interface module. Per the specification [17], the MicroLogix 1000 implementation of the command provides information including firmware, processor mode, and processor random access memory (RAM) size for the interface (24 bytes [17]). Documentation specific to the MicroLogix 1100 implementation of the command is not available. This command has no input parameter to fuzz, and thus, it is only functionally tested to determine MicroLogix 1100-specific responses. Figures 46 and 47 show the structure of an example packet.

```

###[ ENIP TCP ]###
    Command = Send Unit Data (0x0070)
    Length = 40
    Session_Handle= 0xf020100
    Status = Success
    Sender_Context= 0
    Options = 0
###[ Send Unit Data ]###
    Interface_Handle= 0
    Timeout = 1
###[ ENIP_CommonPacketFormat ]###
    Item_Count= 2
    \Items \
        ###[ Common Packet Format Item ]###
        Address_Data_Item= Connection-Based (0x00A1)
        Address_Length= 4
        Connection_Identifier= 0x9f9d0b6f
        ###[ Common Packet Format Item ]###
        Address_Data_Item= Connected Transport Packet (0x00B1)
        Data_Length= 20
        Sequence_Number= 0x1
###[ Common_Industrial_Protocol ]###
    Request_Response= Request
    Common_Service= Execute_PCCC_Service
    Request_Path_Size= 2
    \Words \
        ###[ CIP Request Path ]###
        Path_Segment_Type= Logical Segment
        Logical_Segment_Type= Class ID
        Logical_Segment_Format= 8-bit logical address
        Class = 0x67
        ###[ CIP Request Path ]###
        Path_Segment_Type= Logical Segment
        Logical_Segment_Type= Instance ID
        Logical_Segment_Format= 8-bit logical address
        Eight_bit_Instance= 0x1
###[ CIP Execute PCCC Service Request ]###
    Length_of_Requestor_ID= 7
    CIP_Vendor_ID_of_Requestor= Rockwell Software, Inc.
    CIP_Serial_Number= 90180339
    CMD = 0x06
    Status = 0x0
    Transaction_Word= 2
    Function = Diagnostic_Status

```

Fields encapsulated at the PCCC layer are highlighted.

Figure 46. An Example PCCC Diagnostic Status over TCP Request

0000	00	1d	9c	ca	cb	1b	90	e2	ba	18	fc	2e	08	00	45	00
0010	00	68	00	01	00	00	40	06	e4	8a	0a	01	1e	01	0a	01
0020	64	02	f3	7a	af	12	00	00	00	00	00	00	00	00	50	02
0030	20	00	9b	3d	00	00	70	00	28	00	00	01	02	0f	00	00
0040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0050	00	00	01	00	02	00	a1	00	04	00	6f	0b	9d	9f	b1	00
0060	14	00	01	00	4b	02	20	67	24	01	07	4d	00	f3	0a	60
0070	05	06	00	02	00	03										

Figure 47. Hexadecimal View of Example PCCC Diagnostic Status Request over TCP Packet

7. PCCC Read Diagnostic Counters

Per the specification [17], the MicroLogix 1000 implementation of the command is used to read a module's diagnostic timers and counters by requesting up to 244 bytes of data from the programmable read-only memory (PROM) or RAM of an interface module [17]. The specification does not provide any information specific to the MicroLogix 1100 implementation of the command. This command has two input parameters: Address and Size. The Address field is fuzzed between 0x0000 and 0xFFFF with a Size field set to 0x01. The Size field is fuzzed between 0x00 and 0xFF with the Address field set to 0x0000. Figures 48 and 49 illustrate the structure of an example PCCC Read Diagnostic Counters packet.

```

###[ ENIP TCP ]###
    Command    = Send Unit Data (0x0070)
    Length     = 43
    Session_Handle= 0xf020100
    Status      = Success
    Sender_Context= 0
    Options     = 0
###[ Send Unit Data ]###
    Interface_Handle= 0
    Timeout      = 1
###[ ENIP_CommonPacketFormat ]###
    Item_Count= 2
    \Items      \
        ###[ Common Packet Format Item ]###
        Address_Data_Item= Connection-Based (0x00A1)
        Address_Length= 4
        Connection_Identifier= 0x9f9d0b6f
        ###[ Common Packet Format Item ]###
        Address_Data_Item= Connected Transport Packet (0x00B1)
        Data_Length= 23
        Sequence_Number= 0x1
###[ Common_Industrial_Protocol ]###
    Request_Response= Request
    Common_Service= Execute_PCCC_Service
    Request_Path_Size= 2
    \Words      \
        ###[ CIP Request Path ]###
        Path_Segment_Type= Logical Segment
        Logical_Segment_Type= Class ID
        Logical_Segment_Format= 8-bit logical address
        Class      = 0x67
        ###[ CIP Request Path ]###
        Path_Segment_Type= Logical Segment
        Logical_Segment_Type= Instance ID
        Logical_Segment_Format= 8-bit logical address
        Eight_bit_Instance= 0x1
###[ CIP Execute PCCC Service Request ]###
    Length_of_Requestor_ID= 7
    CIP_Vendor_ID_of_Requestor= Rockwell Software, Inc.
    CIP_Serial_Number= 90180339
    CMD      = 0x06
    Status    = 0x0
    Transaction_Word= 2
    Function  = Read Diagnostic Counters
    Address   = 0
    Size      = 0x1

```

Fields encapsulated at the PCCC layer are highlighted.

Figure 48. An Example PCCC Read Diagnostic Counters Request

0000	00	1d	9c	ca	cb	1b	90	e2	ba	18	fc	2e	08	00	45	00
0010	00	6b	00	01	00	00	40	06	e4	87	0a	01	1e	01	0a	01
0020	64	02	f3	7a	af	12	00	00	00	00	00	00	00	00	50	02
0030	20	00	94	3c	00	00	70	00	2b	00	00	01	02	0f	00	00
0040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0050	00	00	01	00	02	00	a1	00	04	00	6f	0b	9d	9f	b1	00
0060	17	00	01	00	4b	02	20	67	24	01	07	4d	00	f3	0a	60
0070	05	06	00	02	00	01	00	00	01							

Figure 49. Hexadecimal View of Example PCCC Read Diagnostic Counters Request Packet

8. PCCC Restart

The PLC Restart command is intended solely for the PLC-3 and is not compatible with the MicroLogix family per the specification [17]. The command terminates any upload or download, revokes upload/download privileges, and initializes a PLC-3 restart. This command is tested with a properly formatted command in order to determine MicroLogix 1100 functionality. Figures 50 and 51 illustrate the structure of an example PCCC Restart request packet.

```

###[ ENIP TCP ]###
    Command    = Send Unit Data (0x0070)
    Length     = 40
    Session_Handle= 0xf020100
    Status      = Success
    Sender_Context= 0
    Options     = 0
###[ Send Unit Data ]###
    Interface_Handle= 0
    Timeout      = 1
###[ ENIP_CommonPacketFormat ]###
    Item_Count= 2
    \Items      \
    |###[ Common Packet Format Item ]###
    | Address_Data_Item= Connection-Based (0x00A1)
    | Address_Length= 4
    | Connection_Identifier= 0x9f9d0b6f
    |###[ Common Packet Format Item ]###
    | Address_Data_Item= Connected Transport Packet (0x00B1)
    | Data_Length= 20
    | Sequence_Number= 0x1
###[ Common_Industrial_Protocol ]###
    Request_Response= Request
    Common_Service= Execute_PCCC_Service
    Request_Path_Size= 2
    \Words      \
    |###[ CIP Request Path ]###
    | Path_Segment_Type= Logical Segment
    | Logical_Segment_Type= Class ID
    | Logical_Segment_Format= 8-bit logical address
    | Class      = 0x67
    |###[ CIP Request Path ]###
    | Path_Segment_Type= Logical Segment
    | Logical_Segment_Type= Instance ID
    | Logical_Segment_Format= 8-bit logical address
    | Eight_bit_Instance= 0x1
###[ CIP Execute PCCC Service Request ]###
    Length_of_Requestor_ID= 7
    CIP_Vendor_ID_of_Requestor= Rockwell Software, Inc.
    CIP_Serial_Number= 90180339
    CMD      = 0x0F
    Status    = 0x0
    Transaction_Word= 2
    Function  = Restart

```

Fields encapsulated at the PCCC layer are highlighted.

Figure 50. An Example PCCC Restart Request

0000	00	1d	9c	ca	cb	1b	90	e2	ba	18	fc	2e	08	00	45	00
0010	00	68	00	01	00	00	40	06	e4	8a	0a	01	1e	01	0a	01
0020	64	02	f3	7a	af	12	00	00	00	00	00	00	00	00	50	02
0030	20	00	9b	2d	00	00	70	00	28	00	00	01	02	0f	00	00
0040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0050	00	00	01	00	02	00	a1	00	04	00	6f	0b	9d	9f	b1	00
0060	14	00	01	00	4b	02	20	67	24	01	07	4d	00	f3	0a	60
0070	05	0f	00	02	00	0a										

Figure 51. Hexadecimal View of Example PCCC Restart Request Packet

9. PCCC Download Completed

The Download Completed command returns a processor to its previous mode upon completion of a complete system download [17]. This command is not intended for the MicroLogix PLC family. Functionality testing is conducted to observe MicroLogix 1100 responses to an illegal command. Figures 52 and 53 illustrate the structure of an example PCCC Download Completed packet.


```

###[ ENIP TCP ]###
    Command    = Send Unit Data (0x0070)
    Length     = 40
    Session_Handle= 0xf020100
    Status     = Success
    Sender_Context= 0
    Options    = 0
###[ Send Unit Data ]###
    Interface_Handle= 0
    Timeout       = 1
###[ ENIP_CommonPacketFormat ]###
    Item_Count= 2
    \Items
    \
    ###[ Common Packet Format Item ]###
    | Address_Data_Item= Connection-Based (0x00A1)
    | Address_Length= 4
    | Connection_Identifier= 0x9f9d0b6f
    | ###[ Common Packet Format Item ]###
    | | Address_Data_Item= Connected Transport Packet (0x00B1)
    | | Data_Length= 20
    | | Sequence_Number= 0x1
    | ###[ Common_Industrial_Protocol ]###
    | Request_Response= Request
    | Common_Service= Execute_PCCC_Service
    | Request_Path_Size= 2
    | \Words
    | | ###[ CIP Request Path ]###
    | | | Path_Segment_Type= Logical Segment
    | | | Logical_Segment_Type= Class ID
    | | | Logical_Segment_Format= 8-bit logical address
    | | | Class = 0x67
    | | | ###[ CIP Request Path ]###
    | | | Path_Segment_Type= Logical Segment
    | | | Logical_Segment_Type= Instance ID
    | | | Logical_Segment_Format= 8-bit logical address
    | | | Eight_bit_Instance= 0x1
    | ###[ CIP Execute PCCC Service Request ]###
    | Length_of_Requestor_ID= 7
    | CIP_Vendor_ID_of_Requestor= Rockwell Software, Inc.
    | CIP_Serial_Number= 90180339
    | CMD = 0x0F
    | Status = 0x0
    | Transaction_Word= 2
    | Function = Download Completed

```

Fields encapsulated at the PCCC layer are highlighted.

Figure 52. An Example PCCC Download Completed Request

0000	00	1d	9c	ca	cb	1b	90	e2	ba	18	fc	2e	08	00	45	00
0010	00	68	00	01	00	00	40	06	e4	8a	0a	01	1e	01	0a	01
0020	64	02	f3	7a	af	12	00	00	00	00	00	00	00	00	50	02
0030	20	00	9a	e5	00	00	70	00	28	00	00	01	02	0f	00	00
0040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0050	00	00	01	00	02	00	a1	00	04	00	6f	0b	9d	9f	b1	00
0060	14	00	01	00	4b	02	20	67	24	01	07	4d	00	f3	0a	60
0070	05	0f	00	02	00	52										

Figure 53. Hexadecimal View of Example PCCC Download Completed Request Packet

10. PCCC Protected Logical Read with Three Address Fields Command on ControlLogix

The Protected Logical Read with Three Address Fields is tested on the ControlLogix PLC to address this thesis' secondary research question: whether vulnerabilities discovered on earlier model AB/RA PLCs affect more advanced and modern AB/RA PLCs. Previous ENIP Fuzz testing led to the discovery of a vulnerability in MicroLogix's implementation of the command. When any combination of a File Number 0x2 to 0x8 and File Type of 0x47 or 0x48 is present in the command, the MicroLogix 1100 experiences a Major Error (0x8) and enters a fault state [9].

To test the ControlLogix, the fuzzer sends Protected Logical Read with Three Address Field commands with a File Number between 0x2 and 0x8 and File Type of 0x47 or 0x48 to determine if the ControlLogix is susceptible to the same vulnerability affecting MicroLogix PLCs. Figures 54 and 55 illustrate the structure of an example PCCC Download Completed packet.


```

###[ ENIP TCP ]###
    Command    = Send Unit Data (0x0070)
    Length     = 45
    Session_Handle= 0xf020100
    Status     = Success
    Sender_Context= 0
    Options    = 0
###[ Send Unit Data ]###
    Interface_Handle= 0
    Timeout     = 1
###[ ENIP_CommonPacketFormat ]###
    Item_Count= 2
    \Items
    \###[ Common Packet Format Item ]###
        Address_Data_Item= Connection-Based (0x00A1)
        Address_Length= 4
        Connection_Identifier= 0x9f9d0b6f
    \###[ Common Packet Format Item ]###
        Address_Data_Item= Connected Transport Packet (0x00B1)
        Data_Length= 25
        Sequence_Number= 0x1
###[ Common_Industrial_Protocol ]###
    Request_Response= Request
    Common_Service= Execute_PCCC_Service
    Request_Path_Size= 2
    \Words
    \###[ CIP Request Path ]###
        Path_Segment_Type= Logical Segment
        Logical_Segment_Type= Class ID
        Logical_Segment_Format= 8-bit logical address
        Class = 0x67
    \###[ CIP Request Path ]###
        Path_Segment_Type= Logical Segment
        Logical_Segment_Type= Instance ID
        Logical_Segment_Format= 8-bit logical address
        Eight_bit_Instance= 0x1
###[ CIP Execute PCCC Service Request ]###
    Length_of_Requestor_ID= 7
    CIP_Vendor_ID_of_Requestor= Rockwell Software, Inc.
    CIP_Serial_Number= 90180339
    CMD = 0x0F
    Status = 0x0
    Transaction_Word= 1
    Function = Protected_Typed_Logical_Read_Three_Address_Fields
    Byte_Size = 0x1
    File_No = 0x1
    File_Type = 0x1
    Element_No= 0x1
    Sub_Element_No= 0x0

```

Fields encapsulated at the PCCC layer are highlighted.

Figure 54. An Example PCCC Protected Logical Read with Three Address Fields Request

0000	00	1d	9c	ca	cb	1b	90	e2	ba	18	fc	2e	08	00	45	00
0010	00	6d	00	01	00	00	40	06	e4	85	0a	01	1e	01	0a	01
0020	64	02	f3	7a	af	12	00	00	00	00	00	00	00	00	50	02
0030	20	00	8e	8f	00	00	70	00	2d	00	00	01	02	0f	00	00
0040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0050	00	00	01	00	02	00	a1	00	04	00	6f	0b	9d	9f	b1	00
0060	19	00	01	00	4b	02	20	67	24	01	07	4d	00	f3	0a	60
0070	05	0f	00	01	00	a2	01	01	01	01	01	00				

Figure 55. Hexadecimal View of Example PCCC Protected Logical Read with Three Address Fields Request Packet

THIS PAGE INTENTIONALLY LEFT BLANK

V. TEST ANALYSIS

This chapter presents the results of the fuzzed commands and a detailed analysis for each test case examined. Results are summarized first and subsequently expanded upon in the individual command result sections. Wireshark captures of SUT responses are included in Appendixes A through C.

A. ENIP TEST RESULTS

The ENIP tests do not cause any faults or disruption of service to the MicroLogix SUT. However, the testing does reveal several instances where the MicroLogix implementation of ENIP deviates from the specification [16]. Table 9 summarizes both expected and observed responses to the test cases.

Table 9. ENIP Fuzz Testing Results

Test Number	ENIP Command	Fuzzed Field	Protocol	Expected Fuzzed Response	Actual Fuzzed Response
T1	List Services/Identity/Interfaces	Session Handle	TCP	Session Handle repeated in response (ignored by target)	Session Handle repeated in response (ignored by target)
T2	List Services/Identity/Interfaces	Session Handle	UDP	Session Handle repeated in response	Session Handle repeated in response
T3	List Services/Identity/Interfaces	Status	TCP	TCP ACK	TCP ACK
T4	List Services/Identity/Interfaces	Status	UDP	No response	No response
T5	List Services/Identity/Interfaces	Sender Context	TCP	Sender Context repeated in response	Sender Context repeated in response
T6	List Services/Identity/Interfaces	Sender Context	UDP	Sender Context repeated in response	Sender Context repeated in response

Test Number	ENIP Command	Fuzzed Field	Protocol	Expected Fuzzed Response	Actual Fuzzed Response
T7	List Services/Identity/Interfaces	Options	TCP	Packet discarded	Deviation: See ListServices Results section
T8	List Services/Identity/Interfaces	Options	UDP	Packet discarded	Deviation: See ListServices Results section
T9	UnRegisterSession	Session Handle	TCP	Error 0x03 TCP close	Deviation: Error 0x03 No TCP close
T10	UnRegisterSession	Status	TCP	Error 0x03 TCP close	Deviation: Error 0x03 No TCP close
T11	UnRegisterSession	Sender Context	TCP	TCP close	TCP close
T12	UnRegisterSession	Options	TCP	Error 0x03 TCP close	Deviation: Error 0x03, no TCP close
T13	UnRegisterSession UDP Functionality	N/A	UDP	Error 0x01	Error 0x01
T14	SendRRData	Session Handle	TCP	Error 0x03	Deviation: See SendRRData Results Section
T15	SendRRData	Status	TCP	TCP ACK	Deviation: See SendRRData Results Section
T16	SendRRData	Sender Context	TCP	Successful Response with Sender Context returned	Successful Response with Sender Context returned
T17	SendRRData	Options	TCP	TCP ACK	Deviation: See SendRRData Results Section
T18	SendRRData	Interface Handle	TCP	Error 0x03	Error 0x03

Test Number	ENIP Command	Fuzzed Field	Protocol	Expected Fuzzed Response	Actual Fuzzed Response
T19	SendRRData	TimeOut	TCP	Error 0x03	Deviation: See SendRRData Results Section
T20	SendUnitData	Session Handle	TCP	Error 0x03	Deviation: See SendUnitData Results Section
T21	SendUnitData	Status	TCP	TCP ACK	Deviation: See SendUnitData Results Section
T22	SendUnitData	Sender Context	TCP	Successful Response with Sender Context returned	Successful Response with Sender Context returned
T23	SendUnitData	Options	TCP	TCP ACK	Deviation: See SendUnitData Results Section
T24	SendUnitData	Interface Handle	TCP	Error 0x03	Error 0x03
T25	SendUnitData	TimeOut	TCP	Error 0x03	Deviation: See SendUnitData Results Section
T26	Reserved for Legacy	Command Field	TCP	Error 0x03 or success	Error 0x03 or success
T27	Reserved for Legacy	Command Field	UDP	Error 0x01 or success	Error 0x01 or success
T28	Reserved for Future Use	Command Field	TCP	Error 0x03	Error 0x03
T29	Reserved for Future Use	Command Field	UDP	Error 0x01	Error 0x01

1. ENIP ListServices Results

a. T1 and T2 Test Cases

The SUT responds as expected to the ListServices commands (TCP and UDP) with non-zero values in the Session Handle field, i.e., by ignoring the specified session handle and returning the same session handle for the established session in the response.

b. T3 and T4 Test Cases

When non-zero values are sent in the Status field, the SUT responds as expected, i.e., by returning a TCP FIN, ACK if the command is sent over TCP and dropping the packet if the command is sent over UDP.

c. T5 and T6 Test Cases

The SUT responds predictably to the ListServices commands with fuzzed Sender Context fields over both TCP and UDP, i.e., by returning the same Sender Context value in the response.

d. T7 and T8 Test Cases

Per the ENIP specification, receivers must discard any ENIP ListServices packets with non-zero values in the Options field [16]. For both TCP and UDP, the SUT does not discard the ListServices command with a non-zero value in the Options field, but sends a ENIP response with a 0x03 “Incorrect data” [16] error code.

2. ENIP UnRegisterSession Results

a. T9 Test Case

For the Session Handle field, the MicroLogix implementation of the UnRegisterSession command returns a 0x03 “Incorrect data” [16] response and does not terminate the TCP connect as expected. This is a deviation from the specification that dictates a “receiver shall not reject the UnRegisterSession due to unexpected values in the encapsulation header,” including invalid Session Handles and non-zero Status inputs [16].

b. T10 Test Case

When the UnRegisterSession command is fuzzed with invalid Status codes and valid Session Handles, the MicroLogix returns a 0x03 “Incorrect data” [16] error and the TCP connection is not terminated. This is a deviation from the specification observed in T9 with invalid Session Handle inputs.

c. T11 Test Case

When fuzzing the Sender Context field, the MicroLogix implementation of the UnRegisterSession command returns expected responses and terminates the TCP connection. This complies with the ENIP requirement that receivers do not reject UnRegisterSession commands with unexpected values in the encapsulation header but close the underlying TCP connection instead [16].

d. T12 Test Case

When the Options field is set to a non-zero number, the SUT returns a 0x03 “Incorrect data” [16] response and the TCP connection is not terminated. The specification provides conflicting guidance on the expected behavior SUT behavior. Per the CIP Networks Library: Volume 2 EtherNet/IP Adaptation of CIP specification, “the receiver shall discard packets with a non-zero option field” [16]. The specification also says that the receiver shall not reject UnRegisterSession commands due to “unexpected values in the encapsulation header,” including non-zero Options and that the TCP connection shall be terminated [16].

In order to confirm the TCP session is not closed by UnRegisterSession request with an invalid Options field, an additional test is conducted. Following an UnRegisterSession command with a fuzzed Options field, a CIP Forward Open command is sent to the PLC. The SUT responds to the request with “Success” packet, confirming that the session remains open.

e. T13 Test Case

The MicroLogix complies with the ENIP requirement that an UnRegisterSession command sent over UDP shall be rejected with an 0x01 “Invalid or Unsupported” [16] error code.

3. ENIP SendRRData Results

There are multiple fields where the AB/RA MicroLogix’s implementation of the ENIP protocol deviates from the expected responses derived from the CIP Networks Library: Volume 2 EtherNet/IP Adaptation of CIP [16] specification. The expected and observed behaviors of each fuzzed field are discussed below.

a. T14 Test Case

The Session Handle is returned by the target in the ENIP Register Session reply packet, and is to be used in subsequent encapsulation commands within the ENIP session. When tested with session handles other than the valid handle of the ENIP session, the reply is not a 0x03 “Incorrect data” [16] error code as expected, but a successful service response. The CIP data in the response is identical to a message with a valid session handle (see Appendix A). However, the Wireshark protocol analyzer does not properly format the CIP Connection Manager data in replies to the invalid session handles. This is hypothesized to be a result of Wireshark attempting to match request/reply packet pairs with valid session handles.

b. T15 Test Case

The Status field indicates whether a receiver successfully executes a command. A zero response indicates success. Any other responses correlate to general error codes. According to the ENIP specification, the receiver must ignore all ENIP requests with a non-zero Status field, i.e., does not return a reply [16]. When testing the Status field of the SendRRData command, requests with Status fields between 0x00000000 and 0x0000FFFF are accepted and the SUT provides a successful ENIP-encapsulated CIP response with Status code 0x00000000. This deviates from the specification requirements [16]. When requests are sent with Status codes between 0x00010000 and 0xFFFFFFFF,

the SUT performs as expected and provides no ENIP or ENIP-encapsulated CIP response. Only a TCP ACK packet is sent from the SUT to the fuzzer.

c. T16 Test Case

The Sender Context field allows a sender to place any data in the field. The receiver returns the same data in its response, which can be used by the sender to match requests with their replies [16]. For all tests, the returned values of this field match the expected values.

d. T17 Test Case

The Options field allows a sender to provide additional information about the command [16]. For the SendRRData Request, the specification dictates that the Options field be set to zero, and that the “receiver shall discard any packets with a non-zero option field” [16]. When tested with different non-zero options, the SUT returns successful replies, i.e., the returned status is 0x00000000.

e. T18 Test Case

The Interface Handle field identifies the intended communications interface of the command and must be set to zero for the SendRRData request [16]. When this field is set to a non-zero value, the SUT returns an ENIP response with the error code 0x03, as expected.

f. T19 Test Case

The Timeout field indicates the number of seconds the requested operation shall persist until it expires. When the field is set to zero, the timeout of the ENIP protocol assumes the timeout of the encapsulated protocol (CIP). When encapsulating CIP, the sender must set the Timeout field to zero and the receiver is to ignore the field [16]. The expected result for testing non-zero inputs in the Timeout field with CIP encapsulation is an ENIP response with the 0x03 “Incorrect data” [16] error code. However, the SUT returns successful ENIP-encapsulated CIP responses with the Timeout field set to 1024.

4. ENIP SendUnitData Results

Similar deviations from specification observed with SendRRData testing are also present in the SendUnitData testing.

a. T20 to T23 Test Cases

The fuzzing of Session Handle, Status, and Options fields demonstrate the same unexpected behavior observed in the SendRRData responses described above.

b. T24 Test Case

When the Interface Handle field is set to a non-zero value, the SUT returns an ENIP response with the 0x03 “Incorrect data” [16] error code, as expected.

c. T25 Test Case

The testing of the Timeout field shows unexpected behavior. The expected result for testing non-zero inputs in the Timeout field with CIP encapsulation is an ENIP response with a 0x03 “Incorrect data” [16] error code. However, the SUT returns successful ENIP-encapsulated CIP responses with the Timeout field set to zero. The unexpected SendUnitData responses are different than the unexpected SendRRData responses fuzzed under the same conditions. SendUnitData returns successful responses with the Timeout field set to zero, whereas SendRRData returns successful responses with the Timeout field set to 1024.

5. ENIP Reserved for Legacy Use Results

a. T26 and T27 Test Cases

The expected responses for the Legacy Use commands over TCP and UDP are a successful response, a 0x03 “Incorrect data” [16] response, or a 0x01 “Invalid or Unsupported” [16] response. Without knowledge of the packet structure for the Legacy Use commands, testing is limited to test packets that only include the individual Legacy Use command with no additional data attached. All commands sent over TCP return ENIP responses with the error code 0x03, except for the command code 0x01, which returns a successful ENIP response with the SUT’s IP address in the data field. The

Wireshark dissector recognized the commands 0x72 and 0x73 as Indicate Status and Cancel, respectively [16]. UDP-sent commands behave similarly to TCP-sent commands with regards to returning a successful response to command code 0x01. For all other command codes, UDP-sent commands returned the error code 0x01.

6. ENIP Reserved for Future Use Results

a. T28 and T29 Test Cases

Responses to the Future Use commands are expected to be an error code, either 0x03 or 0x01. The SUT returns the error code 0x03 status codes for TCP test cases and the error code 0x01 for UDP test cases.

B. CIP TEST RESULTS

The CIP Fuzzing tests do not cause any faults or disruption of service to the MicroLogix SUT. The test results indicate that MicroLogix does not support several of the tested commands. Table 10 summarizes both the expected and observed responses to the test cases.

Table 10. CIP Fuzz Testing Results

Test Number	CIP Command	Fuzzed Field	Expected Fuzzed Response	Actual Fuzzed Response
T30	Get_Attributes_All	Class	Class specific	Class specific See results below
T31	Get_Attributes_All	Instance	Attribute or Path destination unknown responses	Attribute or Path destination unknown responses
T32	Get_Attribute_List	Class	Attribute, Service not supported, or Path destination unknown responses	Service not supported or Path destination unknown responses
T33	Get_Attribute_List	Attribute_list	Attribute, Service not supported, or Path destination unknown responses	Service not supported or Path destination unknown responses

Test Number	CIP Command	Fuzzed Field	Expected Fuzzed Response	Actual Fuzzed Response
T34	Get_Attribute_List	Instance	Attribute, Service not supported, or Path destination unknown responses	Service not supported or Path destination unknown responses
T35	Get_Attribute_List	Attribute_count	Error status or no response for Attribute_count fields exceeding maximum allowable	TCP ACK for values greater than 223 Attributes in Attribute_count
T36	Get_Attribute_Single	Class	Class specific	Class specific See results below
T37	Get_Attribute_Single	Instance	Attribute not supported or Service not supported	Service not supported, Attribute not supported, or Path destination unknown
T38	Get_Attribute_Single	Attribute	Service not supported	Service not supported or Attribute not supported
T39	Find_Next_Object_Instance	Class	Service not supported or Path destination unknown	Service not supported or Path destination unknown
T40	Find_Next_Object_Instance	Instance	Service not supported or Path destination unknown	Service not supported or Path destination unknown
T41	Find_Next_Object_Instance	Maximum Returned Values	Service not supported	Service not supported

1. CIP Get_Attributes_All Results

To determine a baseline MicroLogix response for the Get_Attributes_All command, a packet with Class 0x01 (Identity) and Instance 0x01 is sent to the SUT. All CIP devices are required to support Instance 0x01 of the Identity Object [15]. The MicroLogix returns a successful CIP response with the seven required attributes: Vendor ID, Device Type, Product Code, Major and Minor Revisions, Status, Serial Number, and

Product Name (see Table 11). The thirteen optional or conditional attributes defined in the specification are not observed in the MicroLogix responses [15].

Table 11. Identity Object Instance Attributes. Adapted from [15].

Attr ID	Need in Implem	Access Rule	NV	Name	Data Type	Description of Attribute
1	Required	Get	NV	Vendor ID	UINT	Identification of each vendor by number
2	Required	Get	NV	Device Type	UINT	Indication of general type of product
3	Required	Get	NV	Product Code	UINT	Identification of a particular product of an individual vendor
4	Required	Get	NV	Revision	STRUCT of:	Revision of the item the Identity Object represents
				Major Revision	USINT	
				Minor Revision	USINT	
5	Required	Get	V	Status	WORD	Summary status of device
6	Required	Get	NV	Serial Number	UDINT	Serial number of device
7	Required	Get	NV	Product Name	SHORT_STRING	Human readable identification

a. T30 Test Case

When the Class field is fuzzed with values from 0x0 to 0xFF, with Instance 0x01, the following behavior was observed:

- Three different Class field inputs between 0x00 and 0xFF return successful CIP packets with attribute information: 0x01 (Identity), 0xF5 (TCP/IP Interface), and 0xF6 (Ethernet Link).
- Three Class field inputs return General Status 0x08 “Service not supported” [15] responses: 0x02 (Message Router), 0x06 (Connection Manager), and 0x67 (PCCC Object).
- The remaining Class inputs return CIP responses with General Status 0x05 “Path destination unknown” [15]. This code is used when the target device does not recognize a class, instance or structure element in the object’s request [15].

b. T31 Test Case

When fuzzing the Instance field with values from 0x00 to 0xFF with the Class field set to 0x01, the SUT responds successfully to two Instance inputs: 0x00 and 0x01. The responses for all other Instance inputs indicate a General Status 0x05 “Path destination unknown” [15]. This is an expected response.

The Instance 0x00 is handled as a special case because it references the Class instead of a particular Instance within the class [15]. Therefore, the response of the Instance 0x00 is at the Class level as shown in Table 12.

Table 12. Identity Object Get_Attributes_All Response for Instance 0x00.
Source: [15].

Attribute ID	Data Type	Attribute Name	Default Value (if not implemented)
1	UINT	Revision	1
2	UINT	Max Instance	1
6	UINT	Max ID Number Class Attributes	0
7	UINT	Max ID Number Instance Attributes	0

The other successful response, Instance 0x01, is used as a baseline command and is previously explained.

2. CIP Get_Attribute_List Results

To determine baseline functionality, a request is sent to the SUT with the following parameters: Class 0x01, Instance 0x01, and Attribute 0x01. There are two possible expected SUT responses. If the SUT supports the command, it is to respond with the requested Attribute (Vendor ID) information. However, since Get_Attribute_List is an optionally supported command at the Class and Instance level [15], the SUT may not provide the requested Attribute response. When tested, the MicroLogix responds with a General Status 0x08 “Service not supported” [15] packet.

a. T32 Test Case

To test the Class field with values 0x00 to 0xFF, the Instance and Attribute fields are set to 0x01 while Class is fuzzed. The SUT returns a General Status 0x08 “Service not supported” [15] CIP response for six of the Class field inputs: 0x01 (Identity), 0x02 (Message Router), 0x06 (Connection Manager), 0x67 (PCCC Object), 0xF5 (TCP/IP Interface), and 0xF6 (Ethernet Link). All other responses have a General Status of 0x05 “Path destination unknown” [15].

b. T33 Test Case

While testing the Instance field with values 0x00 to 0xFF, the Class and Attribute fields are set to 0x01. Only Instances 0x00 and 0x01 return General Status 0x08 “Service not supported” [15] responses. All other tested Instances return General Status 0x05 “Path destination unknown” [15] responses.

c. T34 Test Case

When fuzzing the Attribute field with the Class and Instance fields set 0x01, the SUT returns General Status 0x08 “Service not supported” [15] responses for each Attribute tested. The tested Attribute values are 0x00 to 0xFF.

d. T35 Test Case

To determine the effects of exceeding the maximum number of attributes that can be requested, packets with increasing Attribute_Count are sent to the SUT. Attribute IDs 1 through 7 are utilized and repeated due to their observed presence from the Get_Attributes_All response previously conducted. The SUT returns CIP responses with Status 0x08 “Service not supported” [15] for Get_Attribute_List requests with Attribute_Counts from 0 to 223. When the SUT receives a Get_Attribute_List request with an Attribute_Count of 224 or greater, it does not send a CIP response. The SUT sends only a TCP ACK in response.

3. CIP Get_Attribute_Single Results

The Get_Attribute_Single request is an optional command and thus it is hypothesized that the response would be a ‘Service not supported’ [15] message. Per the specification [15], the Identity Object only supports this command if Class Attributes are implemented. The observed response from the Get_Attributes_All tests for the Identity Object with Instance 0x00 and Attribute 0x01 return default values, indicating no Class Attributes are set for the Identity Object.

a. T36 Test Case

To test the Class field, the Instance field is set to 0x00 and the Attribute field is set to 0x01. The SUT returns a “Service not supported” CIP response for six of the Class field inputs: 0x01 (Identity), 0x02 (Message Router), 0x06 (Connection Manager), 0x67 (PCCC Object), 0xF5 (TCP/IP Interface), and 0xF6 (Ethernet Link). All other responses have a General Status 0x05 “Path destination unknown” [15].

b. T37 Test Case

When testing the Instance field, the Class and Attribute fields are set to 0x01. The SUT returns “Service not supported” messages when it receives request packets with the Instance field set to 0x00. When it receives requests with Instance 0x01, the SUT returns the General Status 0x14 “Attribute not supported” [15] messages. All other Instances returned General Status 0x05 “Path destination unknown” [15] responses.

c. T38 Test Case

To test the Attribute field, packets are sent with the Class and Instance fields set to 0x01. The SUT responds to all fuzzed Attribute inputs with the General Status 0x14 “Attribute not supported” [15] messages.

4. CIP Find_Next_Object_Instance Results

In order to establish baseline behavior for the Find_Next_Object_Instance request, test packets with Class 0x01 (Identity) and Instance 0x00 fields are sent to the SUT. The Identity Object conditionally supports the command if non-consecutive

Instances exit [15]. From the Get_Attributes_All test using the Identity Object, no non-consecutive Instances are observed. Therefore, our expected and observed behavior of the SUT is to return a General Status 0x08 “Service not supported” [15] message.

a. T39 Test Case

To test the Class field, the Instance field is set to 0x00 while Class is fuzzed. The SUT returns a General Status 0x08 “Service not supported” [15] CIP response for six of the Class field inputs: 0x01 (Identity), 0x02 (Message Router), 0x06 (Connection Manager), 0x67 (PCCC Object), 0xF5 (TCP/IP Interface), and 0xF6 (Ethernet Link). All other responses have a General Status 0x05 “Path destination unknown” [15].

b. T40 Test Case

When testing the Instance field, Class is set to 0x01. Requests with Instance 0x00 and 0x01 return General Status 0x08 “Service not supported” [15] responses. All other fuzzed Instance inputs return General Status 0x05 “Path destination unknown” [15] messages.

c. T41 Test Case

To test the Maximum Returned Values field, Class is set to 0x01 and Instance is set to 0x00. The Maximum Returned Values field is tested with inputs between 0x00 and 0xFF. All requests return General Status 0x08 “Service not supported” [15] responses.

C. PCCC TEST RESULTS

The PCCC tests do not cause any faults or disruption of service to the MicroLogix 1100 (T42-T72) or ControlLogix 1756-L71 (T73) SUTs. Table 13 summarizes both expected and observed responses to the test cases. The N/A indicator in the Fuzzed Field column indicates the command is functionally tested only.

Table 13. PCCC Fuzz Testing Results

Test Number	PCCC Command	Fuzzed Field	Expected Fuzzed Response	Actual Fuzzed Response
MicroLogix Tests				
T42	Echo	Data: 0 bytes	Response with 0 bytes attached	Response with 0 bytes attached
T43	Echo	Data: Max Length	243-byte maximum	247-byte maximum
T44	Echo	Data: 8 bytes	Response with 8 bytes attached	Response with 8 bytes attached
T45	Echo	Data: 9 bytes	Response with 9 bytes attached	Response with 9 bytes attached
T46	Echo	Data: 10 bytes	Response with 10 bytes attached	Response with 10 bytes attached
T47	Echo	Data: 40 bytes	Response with 40 bytes attached	Response with 40 bytes attached
T48	Echo	Data: 243 bytes	Response with 243 bytes attached	Response with 243 bytes attached
T49	Echo	Data: Maximum bytes returned by module with no errors	Response with same number of bytes attached as request	Response with 247 bytes attached
T50	Echo	Data: 248 bytes	Response with error message	“Routing failure, request packet too large” [17] response
T51	Echo	Data: 256 bytes	Response with error message	“Routing failure, request packet too large” [17] response
T52	Protected Typed File Read	Size	Response with requested data or error message	“illegal command or format” [17] response
T53	Protected Typed File Read	Tag	Response with requested data or error message	“illegal command or format” [17] response
T54	Protected Typed File Read	Offset	Response with requested data or error message	“illegal command or format” [17] response

Test Number	PCCC Command	Fuzzed Field	Expected Fuzzed Response	Actual Fuzzed Response
T55	Protected Typed File Read	File Type	Response with requested data or error message	“illegal command or format” [17] response
T56	Protected Typed File Write	Size	Response with no errors or error message	“illegal command or format” [17] response
T57	Protected Typed File Write	Tag	Response with no errors or error message	“illegal command or format” [17] response
T58	Protected Typed File Write	Offset	Response with no errors or error message	“illegal command or format” [17] response
T59	Protected Typed File Write	File Type	Response with no errors or error message	“illegal command or format” [17] response
T60	Protected Typed File Write	Data	Response with no errors or error message	“illegal command or format” [17] response
T61	Protected Typed Logical Write with Three Address Fields	Size	Response with no errors or error message	“illegal command or format” [17] or “access denied, improper privilege” [17] responses
T62	Protected Typed Logical Write with Three Address Fields	File No.	Response with no errors or error message	“illegal command or format” [17] response
T63	Protected Typed Logical Write with Three Address Fields	File Type	Response with no errors or error message	“illegal command or format” [17] response
T64	Protected Typed Logical Write with Three Address Fields	Element No.	Response with no errors or error message	“illegal command or format” [17] response
T65	Protected Typed Logical Write with Three Address Fields	Sub-Element No.	Response with no errors or error message	“illegal command or format [17] response
T66	Unprotected Read	Address	Response with requested data or error message	“illegal command or format” [17] response

Test Number	PCCC Command	Fuzzed Field	Expected Fuzzed Response	Actual Fuzzed Response
T67	Unprotected Read	Size	Response with requested data or error message	“illegal command or format” [17] response
T68	Diagnostic Status-Functionality Test	N/A	Diagnostic Status information response	Diagnostic Status information response
T69	Read Diagnostic Counters	Address	Response with requested data or error message	“illegal command or format” [17] response
T70	Read Diagnostic Counters	Size	Response with requested data or error message	“illegal command or format” [17] response
T71	Restart-Functionality Test	N/A	Response with error message	“illegal command or format” [17] response
T72	Download Completed-Functionality Test	N/A	Response with error message	“access denied, improper privilege” [17] response
ControlLogix Tests				
T73	Protected Typed Logical Read with Three Address Fields	File No., File Type	SUT Fault	No fault. EXT STS “Address doesn’t point to something usable” [17] response

1. PCCC Echo Results

a. T42 to T51 Test Cases

The SUT returns successful responses to properly formatted PCCC Echo requests. The data specified in Echo requests, up to 247 bytes, are successfully transmitted back to the fuzzer in a CIP-encapsulated response packet. The observed 247-byte limit exceeds the maximum of 243 data bytes indicated in the specification [17]. When Echo commands are transmitted with greater than 247 data bytes attached, the SUT returns a CIP-encapsulated response indicating “Routing failure, request packet too large.”

2. PCCC Protected Typed File Read Results

a. T52-T55 Test Cases

The SUT responds uniformly to all fuzzed Size, Tag, Offset, and File Type field inputs by returning a STS 0x10 “illegal command or format” [17] code.

3. PCCC Protected Typed File Write Results

a. T56-T60 Test Cases

The SUT responds to all fuzzed Size, Tag, Offset, File Type, and Data field inputs by returning a STS 0x10 “illegal command or format” [17] code.

4. PCCC Protected Logical Write with Three Address Fields Results

a. T61 Test Case

When fuzzing the Byte Size field of the command, all inputs except 0x00 return successful CIP-encapsulated PCCC packets with a STS 0x10 “illegal command or format” [17] code. When the Byte Size field is set to 0x00, the STS field returns 0xF0, indicating an EXT STS is appended. The returned EXT STS byte is 0x0B, indicating “access denied, improper privilege” [17].

b. T62-T65 Test Cases

The SUT responses uniformly to all fuzzed File Number, File Type, Element Number, and Sub-Element Number field inputs by returning a STS 0x10 “illegal command or format” [17] code.

5. PCCC Unprotected Read Results

a. T66-T67 Test Cases

The SUT responses uniformly to all fuzzed Address and Size field inputs by returning a STS 0x10 “illegal command or format” [17] code.

6. PCCC Diagnostic Status Results

a. T68 Test Case

The Diagnostic Status command returns a successful CIP-encapsulated PCCC response. The specification [17] states that the MicroLogix 1000's response is 24 bytes [17]. The MicroLogix 1100 returns 25 bytes of data. Due to this difference, it is not possible to determine the exact meaning of the returned byte values. It appears that the returned data provides information on the SUT's system status as well as an ASCII representation that displays the SUT's model information: 1763-LEC.

7. PCCC Read Diagnostic Counters Results

a. T69 Test Case

When fuzzing the Address field of the Read Diagnostic Counters command, the SUT returns a CIP-encapsulated PCCC response with a STS 0x10 "illegal command or format" [17] code, for all cases except when the Address field is set to 0x0000. During testing, the Size field is constant at 0x01.

b. T70 Test Case

When fuzzing the Size field, the SUT responds with the requested number of bytes when the Size inputs are below 0x6D. These responses contain bytes with zero and non-zero values. The SUT responds to any input of 0x6D or greater with a packet containing no returned data and a STS 0x10 "illegal command or format" [17] code.

8. PCCC Restart Results

a. T71 Test Case

Responses to the Restart command functionality test have STS 0x10 "illegal command or format" [17] codes.

9. PCCC Download Completed Results

a. T72 Test Case

The SUT responds to the Download Completed command with an EXT STS 0x0B “access denied, privilege violation” [17] code.

10. PCCC Protected Logical Read with Three Address Fields on ControlLogix Results

a. T73 Test Case

We speculate that the MicroLogix vulnerability related to this command [9] would be present in the more advanced ControlLogix PLC due to the common practice of reusing legacy code without proper testing in different products from the same manufacturer. Fuzzing the File No. and File Type fields of the Protected Logical Read with Three Address Fields does not produce a fault in the ControlLogix, as observed in the MicroLogix. This proves our hypothesis false.

There is an observable difference between the MicroLogix and ControlLogix responses to the command when File No. and File Type are fuzzed. From previous testing [1], we observe that MicroLogix responds in one of five ways: 1) responds with an STS 0x10 “illegal command or format” code, 2) responds with an EXT STS 0x0B “Access denied, improper privilege” [17] code, 3) responds with an EXT STS 0x0C “condition cannot be generated, resource is not available” [17] code, 4) responds with data, or 5) responds by entering a fault condition [17]. Table 14 illustrates sample request packet field contents and the range of SUT responses.

Table 14. Example MicroLogix 1100 Responses to PCCC Protected Logical Read with Three Address Fields Command

Byte Size	File Type	File No.	Element No.	Sub-element No.	SUT Response
0x01	0x10	0xD0	0x84	0x00	STS 0x10
0x57	0x75	0x65	0x10	0x00	EXT STS 0x0B
0x56	0xBD	0x4C	0x59	0x00	EXT STS 0x0C
0x1C	0x2A	0x62	0x01	0x00	Data response
0xC8	0x03	0x47	0xBC	0x00	Fault response

In all tests, the ControlLogix SUT returns a STS 0xF0 “Error code in the EXT STS byte” code and an EXT STS byte of 0x06 “Address doesn’t point to something usable” [17]. This difference in SUT responses may be a useful tool in fingerprinting the manufacturer and model of a target PLC.

D. DISCUSSION

Our fuzz testing does not uncover any MicroLogix 1100 vulnerabilities. However, we observe some deviations from the expected responses in the MicroLogix implementation of ENIP and PCCC protocols. No CIP deviations are observed. Multiple optional tested CIP commands are not supported by MicroLogix 1100 PLCs. Table 15 provides a summary of the discovered MicroLogix unexpected responses.

Table 15. Summary of MicroLogix 1100Unexpected Responses

Test Number	Command	Fuzzed Field	Protocol	Expected Fuzzed Response	Deviation Response
ENIP Tests					
T7	List Services/Identity/Interfaces	Options	TCP	Packet discarded	Error 0x03 response
T8	List Services/Identity/Interfaces	Options	UDP	Packet discarded	Error 0x03 response
T9	UnRegisterSession	Session Handle	TCP	Error 0x03 TCP close	Error 0x03 response No TCP close
T10	UnRegisterSession	Status	TCP	Error 0x03 TCP close	Error 0x03 response No TCP close
T12	UnRegisterSession	Options	TCP	Error 0x03 TCP close	Error 0x03 response no TCP close
T14	SendRRData	Session Handle	TCP	Error 0x03	No error, Successful response
T15	SendRRData	Status	TCP	TCP ACK	Successful responses for Status fields between 0x00000000 and 0x0000FFFF

Test Number	Command	Fuzzed Field	Protocol	Expected Fuzzed Response	Deviation Response
T17	SendRRData	Options	TCP	TCP ACK	Successful response
T19	SendRRData	Timeout	TCP	Error 0x03	Successful response, Timeout field 1024
T20	SendUnitData	Session Handle	TCP	Error 0x03	No error, Successful response
T21	SendUnitData	Status	TCP	TCP ACK	Successful responses for Status fields between 0x00000000 and 0x0000FFFF
T23	SendUnitData	Options	TCP	TCP ACK	Successful response
CIP Tests					
No observed deviations from specification: Tested optional commands not implemented by MicroLogix 1100					
PCCC Tests					
T43	PCCC Echo	Data: Max Length	TCP	243-byte maximum	247-byte maximum
T52-T55	Protected Typed File Read	Size, Tag, Offset, File Type	TCP	Data response	“illegal command or format” [17] response
T56-T60	Protected Typed File Write	Size, Tag, Offset, File Type, Data	TCP	Data response	“illegal command or format” [17] response
T61-T65	Protected Typed Logical Write with Three Address Fields	Size, File No., File Type, Element No. Sub-Element No.	TCP	Response with no errors or error message	“illegal command or format” [17] response
T66-T67	Unprotected Read	Address, Size	TCP	Data response	“illegal command or format” [17] response
T68	Diagnostic Status	N/A, Functionality Test	TCP	24-byte Diagnostic Status information response	25-byte Diagnostic Status information response

The deviations in the ENIP implementation may be the result of manufacturer implementation decisions. A potential explanation for the PCCC deviations is that the reference specification [9] applies to the MicroLogix 1000 model. While we expect the implementation to be similar between the 1000 and 1100 models, there are differences in processing capability, memory allocations, and functionality between the PLCs, which may account for the deviations.

Our ControlLogix testing disproves the hypothesis that the PCCC Protected Typed Logical Read with Three Address Fields vulnerability in MicroLogix 1100 also affects the ControlLogix 1756-L71. In contrast to the fault condition observed on the MicroLogix 1100, the ControlLogix 1756-L71 returns an error message upon receiving a request with the File No. field ranges between 0x2 to 0x8 and the File Type is 0x47 or 0x48. Table 16 illustrates the ControlLogix response.

Table 16. Summary of ControlLogix 1756-L71 Response Deviations

Test Number	Command	Fuzzed Field	Protocol	Expected Fuzzed Response	Deviation Response
T73	Protected Typed Logical Read with Three Address Fields	File No. (0x02-0x08), File Type (0x47 or 0x48)	TCP	SUT Fault	No fault. EXT STS 0x06 "Address doesn't point to something usable" [1] response

The deviations may provide useful information for application-layer fingerprinting of PLC devices. By cataloging the unique responses returned from the MicroLogix 1100 and ControlLogix 1756-L71, we can begin compiling a corpus of PLC response signatures. This can be used to classify PLC modules through traffic analysis.

VI. CONCLUSION AND FURTHER WORK

A. SUMMARY

Motivated by the increasing employment of industrial control systems on U.S. Navy vessels and the potential for vulnerabilities in the utilized communication protocols, we aim to test the implementation of industrial network protocols on a PLC. Two hypotheses drive our testing. The first hypothesis is that undiscovered software flaws existed in the implementation of ENIP, CIP, and PCCC protocols used by the MicroLogix PLCs. The second hypothesis is that network vulnerabilities known to exist in older PLCs help inform on the robustness of the ICS network stack in more modern PLCs.

To verify our hypotheses, we use a fuzz testing methodology to stress test selected fields in target commands and monitor the system responses. To accomplish this, we use the Scapy-based ENIP Fuzz program [9] and modify the code to expand the range of testable protocol commands. We test our first hypothesis on the MicroLogix 1100 PLC by selecting a range of commands from the ENIP, CIP, and PCCC protocols that were not previously tested and systematically fuzzed the modifiable fields. Candidate protocol commands are evaluated for fuzzing based on their likelihood of creating a fault condition while not permanently damaging the test PLC or corrupting the functionality of the MicroLogix system.

The results of our fuzz testing do not uncover any new vulnerabilities in the MicroLogix 1100 PLC. However, we observe several unexpected responses in four ENIP commands (List Services/Identity/Interfaces, UnRegisterSession, SendRRData, and SendUnitData), and six PCCC commands (Echo, Protected Typed File Read, Protected Typed File Write, Protected Logical Write with Three Address Fields, Unprotected Read, and Diagnostic Status).

Our second hypothesis is tested by sending to the ControlLogix 1756-L71 specially crafted PCCC Protected Logical Read with Three Address Fields packets that trigger a fault condition in the MicroLogix 1100 [9]. By replicating the fault-inducing

packet configuration of the command and applying it to a more advanced PLC, we aim to test if cross-generational vulnerabilities existed in AB/RA PLCs.

Instead of entering a fault state, the ControlLogix 1756-L71 PLC returns an error message upon receiving the fault-inducing test packets. This behavior disproves our hypothesis that the same MicroLogix 1100 vulnerability would affect the ControlLogix 1756-L71 PLC.

B. FUTURE WORK

In addition to PLC fingerprinting, the unique SUT responses observed during our testing may also be used by an intrusion detection system to catch malicious probing activities. To provide a larger context and differentiation among various PLCs, we plan to perform additional EtherNet/IP fuzz testing on the ControlLogix 1756-L71 and other ControlLogix models. These tests will provide insights on whether the observed response to the PCCC Protected Logical Read with Three Address Field command is specific to that command or is common to all PCCC requests, and on whether the PCCC support is the same or different across ControlLogix models.

Another extension to this work is to test the MicroLogix 1000 PLC to determine if the deviations observed in the MicroLogix 1100 are unique to that model or if the MicroLogix family uses a different implementation than detailed in the specification [17].

The scope of this thesis focuses on two different generations of AB/RA PLCs and the EtherNet/IP protocol suite. The ENIP Fuzz program can be enhanced to support other industrial protocols such as PROFINET or DNP3. The enhancement will provide a flexible test platform, which can be used to perform penetration testing, intrusion detection, and fingerprinting reconnaissance on a wide range of industrial control systems.

APPENDIX A. ENIP COMMAND RESPONSES

The following Wireshark captures in Figures 56–93 illustrate test case responses for each command. For certain test cases, the corresponding request command sent to the SUT is also included to show how select fuzzed field inputs affect SUT responses. For descriptions of SUT responses, see Chapter V: Test Analysis.

A. ENIP LISTSERVICES TEST CASES

This section shows the results of the ENIP ListServices test cases.

(1) T1 Results

No.	Time	Source	Destination	Protocol	Length	Resp	Info
24	20:53:52.705232	192.168.0.62	192.168.0.59	TCP	60	47986 → 44818 [ACK] Seq=1 Ack=1 Win=29200 Len=0	
25	20:53:52.709004	192.168.0.62	192.168.0.59	ENIP	78	List Services (Req)	
26	20:53:52.714639	192.168.0.59	192.168.0.62	ENIP	104	List Services (Resp), Communications	
27	20:53:52.714790	192.168.0.62	192.168.0.59	TCP	60	47986 → 44818 [ACK] Seq=25 Ack=51 Win=29200 Len=0	
28	20:53:52.800005	192.168.0.62	192.168.0.59	TCP	60	47986 → 44818 [RST] Seq=75 Ack=51 Win=29200 Len=0	
▶ Transmission Control Protocol, Src Port: 44818, Dst Port: 47986, Seq: 1, Ack: 25, Len: 50							
▼ Ethernet/IP (Industrial Protocol), Session: 0xFFFFFFFF, List Services							
▼ Encapsulation Header							
Command: List Services (0x0004)							
Length: 26							
Session Handle: 0xffffffff							
Status: Success (0x00000000)							
Sender Context: 0000000000000000							
Options: 0x00000000							
▼ Command Specific Data							
▼ Item Count: 1							
▼ Type ID: List Services Response (0x0100)							
Length: 20							
Encapsulation Version: 1							
▼ Capability Flags: 0x0020							
.... 0001 = Supports CIP Encapsulation via TCP: True							
..... 0 = Supports CIP Class 0 or 1 via UDP: False							
Name of Service: Communications							
0000	00 0c 29 56 20 17 00 1d	9c a1 28 4c 08 00 45 00	..)V(L..E.				
0010	00 5a 00 63 00 00 00 06	b0 71 c0 a0 00 3b c0 a0	..Z.c.... .q...r..				
0020	00 3e af 12 bb 72 4d 66	2c 24 ff 52 1a c0 50 18	..>...rMf ,\$.R..P.				
0030	07 d0 0a d7 00 00 04 00	1a 00 ff ff ff ff 00 00				
0040	00 00 00 00 00 00 00 00	00 00 00 00 00 00 01 00				
0050	00 01 14 00 01 00 20 00	43 6f 6d 6d 75 6e 69 63 Communic				
0060	61 74 69 6f 6e 73 00 00		ations..				

Figure 56. ListServices Response over TCP (Fuzzed Session Handle)

(2) T2 Results

No.	Time	Source	Destination	Protocol	Length	Resp	Info
22	21:12:29.509378	Vmware_56:20:17	Broadcast	ARP	60		Who has 192.168.0.59? Tell 192.168.0.62
23	21:12:29.516296	Rockwell_a1:28:4c	Vmware_56:20:17	ARP	60		192.168.0.59 is at 00:1d:9c:a1:28:4c
24	21:12:29.516159	192.168.0.62	192.168.0.59	ENIP	66		List Services (Req)
25	21:12:29.516539	192.168.0.59	192.168.0.62	ENIP	92		List Services (Resp), Communications
User Datagram Protocol, Src Port: 44818, Dst Port: 51606							
Ethernet/IP (Industrial Protocol), Session: 0xFFFFFFFF, List Services							
Encapsulation Header							
Command: List Services (0x0004)							
Length: 26							
Session Handle: 0xffffffff							
Status: Success (0x00000000)							
Sender Context: 0000000000000000							
Options: 0x00000000							
Command Specific Data							
Item Count: 1							
Type ID: List Services Response (0x0100)							
Length: 20							
Encapsulation Version: 1							
Capability Flags: 0x0020							
.....1..... = Supports CIP Encapsulation via TCP: True							
.....0..... = Supports CIP Class 0 or 1 via UDP: False							
Name of Service: Communications							
0000	00 0c 29 56 20 17 00 1d	9c a1 28 4c 00 00 45 00	..)V(L..E.				
0010	00 4e 04 5e 00 00 00 11	b4 77 c0 a8 00 3b c0 a8	.N.^.... .w....:..				
0020	00 3e af 12 c9 96 00 3a	e7 ff 04 00 1a 00 ff ff	.>.....:				
0030	ff ff 00 00 00 00 00 00	00 00 00 00 00 00 00 00:				
0040	00 00 01 00 00 01 14 00	01 00 20 00 43 6f 6d 6d: ..Comm				
0050	75 6e 69 63 61 74 69 6f	6e 73 00 00	unicatio ns..				

Figure 57. ListServices Response over UDP (Fuzzed Session Handle)

(3) T3 Results

No.	Time	Source	Destination	Protocol	Length	Resp	Info
19	20:40:02.701570	192.168.0.62	192.168.0.59	TCP	60		47946 → 44818 [ACK] Seq=1 Ack=1 Win=29200 Len=0
20	20:40:02.703163	192.168.0.62	192.168.0.59	ENIP	78		List Services (Req)
21	20:40:02.804003	192.168.0.62	192.168.0.59	TCP	60		47946 → 44818 [FIN, ACK] Seq=25 Ack=1 Win=29200 Len=0
22	20:40:02.805051	192.168.0.59	192.168.0.62	TCP	60		44818 → 47946 [ACK] Seq=1 Ack=26 Win=2000 Len=0
Ethernet II, Src: Vmware_56:20:17 (08:0c:29:56:20:17), Dst: Rockwell_a1:28:4c (00:1d:9c:a1:28:4c)							
Internet Protocol Version 4, Src: 192.168.0.62, Dst: 192.168.0.59							
Transmission Control Protocol, Src Port: 47946, Dst Port: 44818, Seq: 25, Ack: 1, Len: 0							
Source Port: 47946							
Destination Port: 44818							
[Stream index: 2]							
[TCP Segment Len: 0]							
Sequence number: 25 (relative sequence number)							
Acknowledgment number: 1 (relative ack number)							
Header Length: 20 bytes							
Flags: 0x011 (FIN, ACK)							
000. = Reserved: Not set							
...0 = Nonce: Not set							
...0 = Congestion Window Reduced (CWR): Not set							
...0 = ECN-Echo: Not set							
...0 = Urgent: Not set							
...1 = Acknowledgment: Set							
...0 = Push: Not set							
...0 = Reset: Not set							
...0 = Syn: Not set							
...1 = Fin: Set							
[Expert Info (Chat/Sequence): Connection finish (FIN)]							
[Connection finish (FIN)]							
[Severity level: Chat]							
[Group: Sequence]							
[TCP Flags:A..F]							
Window size value: 29200							
[Calculated window size: 29200]							
[Window size scaling factor: -2 (no window scaling used)]							
Checksum: 0x0da0 [unverified]							
[Checksum Status: Unverified]							
Urgent pointer: 0							
0000	00 1d 9c a1 28 4c 00 0c	29 56 20 17 00 00 45 00(L..)V ...E.				
0010	00 28 0a 91 40 00 40 06	ae 75 c0 a8 00 3e c0 a8	..(.@.@. .u....>..				
0020	00 3b bb 4a af 12 66 fc	78 4f 4d 59 17 57 50 11	.;J..f. xOMY.WP.				
0030	72 10 0d a0 00 00 00 00	00 00 00 00	f.....:				

Figure 58. ListServices Response over TCP (Fuzzed Status)

(4) T4 Results

No.	Time	Source	Destination	Protocol	Length	Resp	Info
17	21:17:29.046996	Rockwell_a1:28:4c	Vmware_56:20:17	ARP	60		192.168.0.59 is at 00:1d:9c:a1:28:4c
18	21:17:29.064369	192.168.0.62	192.168.0.59	ENIP	66		List Services (Req)
19	21:17:37.529109	Vmware_56:20:17	Broadcast	ARP	60		Who has 192.168.0.59? Tell 192.168.0.62
20	21:17:37.530642	Rockwell_a1:28:4c	Vmware_56:20:17	ARP	60		192.168.0.59 is at 00:1d:9c:a1:28:4c

▶ Frame 18: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
 ▶ Ethernet II, Src: Vmware_56:20:17 (00:0c:29:56:20:17), Dst: Rockwell_a1:28:4c (00:1d:9c:a1:28:4c)
 ▶ Internet Protocol Version 4, Src: 192.168.0.62, Dst: 192.168.0.59
 ▶ User Datagram Protocol, Src Port: 51606, Dst Port: 44818
 ▼ EtherNet/IP (Industrial Protocol), Session: 0x00000000, List Services
 ▼ Encapsulation Header
 Command: List Services (0x0004)
 Length: 0
 Session Handle: 0x00000000
 Status: Unknown (0xffffffff)
 Sender Context: 0000000000000000
 Options: 0x00000000

Figure 59. ListServices Response over UDP (Fuzzed Status)

(5) T5 Results

No.	Time	Source	Destination	Protocol	Length	Resp	Info
4	20:04:39.228610	192.168.0.59	192.168.0.62	TCP	62		44818 → 60392 [SYN, ACK] Seq=0 Ack=1 Win=2000 Len=0 MSS=16384 SACK_PERM=1
7	20:04:39.245078	192.168.0.59	192.168.0.62	ENIP	104		List Services (Rsp), Communications
15	20:05:06.234049	192.168.0.59	192.168.0.62	TCP	62		44818 → 60394 [SYN, ACK] Seq=0 Ack=1 Win=2000 Len=0 MSS=16384 SACK_PERM=1
18	20:05:06.245070	192.168.0.59	192.168.0.62	ENIP	104		List Services (Rsp), Communications
1	20:04:39.226409	192.168.0.62	192.168.0.59	TCP	74		60392 → 44818 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=2694
5	20:04:39.229291	192.168.0.62	192.168.0.59	TCP	60		60392 → 44818 [ACK] Seq=1 Ack=1 Win=29200 Len=0

▶ Frame 18: 104 bytes on wire (832 bits), 104 bytes captured (832 bits) on interface 0
 ▶ Ethernet II, Src: Rockwell_a1:28:4c (00:1d:9c:a1:28:4c), Dst: Vmware_56:20:17 (00:0c:29:56:20:17)
 ▶ Internet Protocol Version 4, Src: 192.168.0.59, Dst: 192.168.0.62
 ▶ Transmission Control Protocol, Src Port: 44818, Dst Port: 60394, Seq: 1, Ack: 25, Len: 50
 ▼ EtherNet/IP (Industrial Protocol), Session: 0x00000000, List Services
 ▼ Encapsulation Header
 Command: List Services (0x0004)
 Length: 26
 Session Handle: 0x00000000
 Status: Success (0x00000000)
 Sender Context: ffffffff
 Options: 0x00000000
 ▶ Command Specific Data
 0000 00 0c 29 56 20 17 00 1d 9c a1 28 4c 00 00 45 00 ..)V(L...E.
 0010 00 5a 00 04 00 00 00 06 b8 d0 c0 a0 00 3b c0 a0 .Z.....:..
 0020 00 3e af 12 eb ea 4d 53 0d 16 77 c9 e1 c9 50 18 .>...MS ..W...P.
 0030 07 d0 b9 ff 00 00 04 00 1a 00 00 00 00 00 00 00
 0040 00 00 ff ff ff ff ff ff ff ff 00 00 00 00 01 00
 0050 00 01 14 00 01 00 20 00 43 6f 6d 6d 75 6e 69 63 Communic
 0060 61 74 69 6f 6e 73 00 00 ations..

Figure 60. ListServices Response over TCP (Fuzzed Sender Context)

(6) T6 Results

No.	Time	Source	Destination	Protocol	Length	Resp	Info
10	21:21:37.151721	192.168.0.62	192.168.0.59	ICMP	120		Destination unreachable (Port unreachable)
11	21:22:06.994622	Vmware_56:20:17	Broadcast	ARP	60		Who has 192.168.0.59? Tell 192.168.0.62
12	21:22:06.996443	Rockwell_a1:28:4c	Vmware_56:20:17	ARP	60		192.168.0.59 is at 00:1d:9c:a1:28:4c
13	21:22:07.055575	192.168.0.62	192.168.0.59	ENIP	66		List Services (Req)
14	21:22:07.056525	192.168.0.59	192.168.0.62	ENIP	92		List Services (Resp), Communications
▶ Frame 14: 92 bytes on wire (736 bits), 92 bytes captured (736 bits) on interface 0 ▶ Ethernet II, Src: Rockwell_a1:28:4c (00:1d:9c:a1:28:4c), Dst: Vmware_56:20:17 (00:0c:29:56:20:17) ▶ Internet Protocol Version 4, Src: 192.168.0.59, Dst: 192.168.0.62 ▶ User Datagram Protocol, Src Port: 44818, Dst Port: 51606 ▼ Ethernet/IP (Industrial Protocol), Session: 0x00000000, List Services ▼ Encapsulation Header Command: List Services (0x0004) Length: 26 Session Handle: 0x00000000 Status: Success (0x00000000) Sender Context: aaaaaaaaaaaaaa Options: 0x00000000 ▼ Command Specific Data ▼ Item Count: 1 ▼ Type ID: List Services Response (0x0100) Length: 20 Encapsulation Version: 1 ▼ Capability Flags: 0x00201..... = Supports CIP Encapsulation via TCP: True0..... = Supports CIP Class 0 or 1 via UDP: False Name of Service: Communications 0000 00 0c 29 56 20 17 00 1d 9c a1 28 4c 00 00 45 00 ..)V(L..E. 0010 00 4e 04 63 00 00 00 11 b4 72 c0 a8 00 3b c0 a8 .N.c.... .r..... 0020 00 3e af 12 c9 96 00 3a 3d 55 94 00 1a 00 00 00 .>.....:u..... 0030 00 00 00 00 00 00 aa aa aa aa aa aa aa aa aa 00 0040 00 00 01 00 00 01 14 00 01 00 20 00 43 6f 6d 6dComm 0050 75 6e 69 63 61 74 69 6f 6e 73 00 00 unication ns..							

Figure 61. ListServices Response over UDP (Fuzzed Sender Context)

(7) T7 Results

No.	Time	Source	Destination	Protocol	Length	Resp	Info
20	20:56:41.833625	192.168.0.59	192.168.0.62	TCP	62		44818 → 47996 [SYN, ACK] Seq=0 Ack=1 Win=2000 Len=0 MSS=16384 SACK_PERM=1
21	20:56:41.834007	192.168.0.62	192.168.0.59	TCP	60		47996 → 44818 [ACK] Seq=1 Ack=1 Win=29200 Len=0
22	20:56:41.836679	192.168.0.62	192.168.0.59	ENIP	78		List Services (Req)
23	20:56:41.840915	192.168.0.59	192.168.0.62	ENIP	78		List Services (Resp)
24	20:56:41.841297	192.168.0.62	192.168.0.59	TCP	60		47996 → 44818 [ACK] Seq=25 Ack=25 Win=29200 Len=0
▼ Ethernet/IP (Industrial Protocol), Session: 0x00000000, List Services ▼ Encapsulation Header Command: List Services (0x0004) Length: 0 Session Handle: 0x00000000 Status: Incorrect Data (0x00000003) Sender Context: 0000000000000000 Options: 0x00000000 0000 00 0c 29 56 20 17 00 1d 9c a1 28 4c 00 00 45 00 ..)V(L..E. 0010 00 40 00 6d 00 00 00 b8 81 c0 a8 00 3b c0 a8 .@.m.... 0020 00 3e af 12 bb 7c 49 21 57 f1 02 cb 2d d6 50 1b .>... E! W...-P. 0030 07 d8 62 d7 00 00 04 00 00 00 00 00 00 03 00 ..b..... 0040 00 00 00 00 00 00 00 00 00 00 00 00 00 00							

Figure 62. ListServices Response over TCP (Fuzzed Options)

(8) T8 Results

No.	Time	Source	Destination	Protocol	Length	Resp	Info
16	21:25:29.628998	Vmware_56:20:17	Broadcast	ARP	60		Who has 192.168.0.59? Tell 192.168.0.62
17	21:25:29.630164	Rockwell_a1:28:4c	Vmware_56:20:17	ARP	60		192.168.0.59 is at 00:1d:9c:a1:28:4c
18	21:25:29.675655	192.168.0.62	192.168.0.59	ENIP	66		List Services (Req)
19	21:25:29.676347	192.168.0.59	192.168.0.62	ENIP	66		List Services (Rsp)
20	21:25:29.676613	192.168.0.62	192.168.0.59	ICMP	94		Destination unreachable (Port unreachable)
▼ Ethernet/IP (Industrial Protocol), Session: 0x00000000, List Services							
▼ Encapsulation Header							
Command: List Services (0x0004)							
Length: 0							
Session Handle: 0x00000000							
Status: Unknown (0x03000000)							
Sender Context: 0000000000000000							
Options: 0x00000000							
0000	00 0c 29 56 20 17 00 1d	9c a1 28 4c 00 00 45 00	..)V(L..E.				
0010	00 34 04 68 00 00 00 11	b4 87 c0 a0 00 3b c0 a0	.4.h.... ..;				
0020	00 3e af 12 c9 96 00 20	01 38 04 00 00 00 00 00	.>..... .8.....				
0030	00 00 00 00 00 03 00 00	00 00 00 00 00 00 00 00				
0040	00 00		..				

Figure 63. ListServices Response over UDP (Fuzzed Options)

B. ENIP UNREGISTERSESSION TEST CASES

This section shows the results of the ENIP UnRegisterSession test cases.

(1) T9 Results

No.	Time	Source	Destination	Protocol	Length	Resp	Info
20	11:05:30.560837	192.168.0.62	192.168.0.59	ENIP	82		Register Session (Req), Session: 0x00000000
21	11:05:30.572069	192.168.0.59	192.168.0.62	ENIP	82		Register Session (Rsp), Session: 0x89566822
22	11:05:30.574646	192.168.0.62	192.168.0.59	TCP	60		48554 → 44818 [ACK] Seq=29 Ack=29 Win=29200 Len=0
23	11:05:30.663874	192.168.0.62	192.168.0.59	ENIP	78		Unregister Session (Req), Session: 0x000000C3
24	11:05:30.671952	192.168.0.59	192.168.0.62	ENIP	78		Unregister Session (Rsp), Session: 0x000000C3
25	11:05:30.672258	192.168.0.62	192.168.0.59	TCP	60		48554 → 44818 [ACK] Seq=53 Ack=53 Win=29200 Len=0
26	11:05:30.765740	192.168.0.62	192.168.0.59	TCP	74		48556 → 44818 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=3213
▶ Frame 24: 78 bytes on wire (624 bits), 78 bytes captured (624 bits) on interface 0							
▶ Ethernet II, Src: Rockwell_a1:28:4c (00:1d:9c:a1:28:4c), Dst: Vmware_56:20:17 (00:0c:29:56:20:17)							
▶ Internet Protocol Version 4, Src: 192.168.0.59, Dst: 192.168.0.62							
▶ Transmission Control Protocol, Src Port: 44818, Dst Port: 48554, Seq: 29, Ack: 53, Len: 24							
▼ Ethernet/IP (Industrial Protocol), Session: 0x000000C3, Unregister Session							
▼ Encapsulation Header							
Command: Unregister Session (0x0066)							
Length: 0							
Session Handle: 0x000000C3							
Status: Incorrect Data (0x00000003)							
Sender Context: 0000000000000000							
Options: 0x00000000							
0000	00 0c 29 56 20 17 00 1d	9c a1 28 4c 00 00 45 00	..)V(L..E.				
0010	00 40 03 01 00 00 00 06	b5 ed c0 a0 00 3b c0 a0	.0..... ..;				
0020	00 3e af 12 bd aa 49 86	bd ab 15 19 aa 9f 50 18	.>...I.P.				
0030	07 d0 c6 71 00 00 66 00	00 00 c3 00 00 00 03 00	...q..f.				
0040	00 00 00 00 00 00 00 00					

Figure 64. UnRegisterSession Response over TCP (Fuzzed Session Handle)

(2) T10 Results

No.	Time	Source	Destination	Protocol	Length	Resp	Info
13	11:24:53.020452	192.168.0.62	192.168.0.59	TCP	60		50984 → 44818 [RST, ACK] Seq=53 Ack=53 Win=29200 Len=0
14	11:24:53.021781	192.168.0.62	192.168.0.59	ENIP	82		Register Session (Req), Session: 0x00000000
15	11:24:53.030279	192.168.0.59	192.168.0.62	ENIP	82		Register Session (Rsp), Session: 0x3108880E
16	11:24:53.030606	192.168.0.62	192.168.0.59	TCP	60		50986 → 44818 [ACK] Seq=29 Ack=29 Win=29200 Len=0
17	11:24:53.124356	192.168.0.62	192.168.0.59	ENIP	78		Unregister Session (Req), Session: 0x3108880E
18	11:24:53.130584	192.168.0.59	192.168.0.62	ENIP	78		Unregister Session (Rsp), Session: 0x3108880E
19	11:24:53.130847	192.168.0.62	192.168.0.59	TCP	60		50986 → 44818 [ACK] Seq=53 Ack=53 Win=29200 Len=0
▶ Frame 18: 78 bytes on wire (624 bits), 78 bytes captured (624 bits) on interface 0							
▶ Ethernet II, Src: Rockwell_a1:28:4c (00:1d:9c:a1:28:4c), Dst: Vmware_56:20:17 (00:0c:29:56:20:17)							
▶ Internet Protocol Version 4, Src: 192.168.0.59, Dst: 192.168.0.62							
▶ Transmission Control Protocol, Src Port: 44818, Dst Port: 50986, Seq: 29, Ack: 53, Len: 24							
▼ EtherNet/IP (Industrial Protocol), Session: 0x3108880E, Unregister Session							
▼ Encapsulation Header							
Command: Unregister Session (0x0066)							
Length: 0							
Session Handle: 0x3108880e							
Status: Incorrect Data (0x00000003)							
Sender Context: 0000000000000000							
Options: 0x00000000							
0000	00 0c 29 56 20 17 00 1d	9c a1 28 4c 00 00 45 00	..)V(L..E.				
0010	00 40 11 3a 00 00 00 06	a7 b4 c0 a0 00 3b c0 a0	.0:.....				
0020	00 3e af 12 c7 2a 4f ca	7f c4 d1 34 d0 63 50 1b	.>...+0. ...4.cP.				
0030	07 d0 ee fb 00 00 66 00	00 00 de 08 00 31 03 00f.1..				
0040	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00				

Figure 65. UnRegisterSession Response over TCP (Fuzzed Status)

(3) T11 Results

No.	Time	Source	Destination	Protocol	Length	Resp	Info
18	11:27:32.792125	192.168.0.62	192.168.0.59	ENIP	82		Register Session (Req), Session: 0x00000000
19	11:27:32.798421	192.168.0.59	192.168.0.62	ENIP	82		Register Session (Rsp), Session: 0x42B48090
20	11:27:32.798783	192.168.0.62	192.168.0.59	TCP	60		51350 → 44818 [ACK] Seq=29 Ack=29 Win=29200 Len=0
21	11:27:32.895004	192.168.0.62	192.168.0.59	ENIP	78		Unregister Session (Req), Session: 0x42B48090
22	11:27:32.898198	192.168.0.59	192.168.0.62	TCP	60		44818 → 51350 [FIN, ACK] Seq=29 Ack=53 Win=2000 Len=0
23	11:27:32.937998	192.168.0.62	192.168.0.59	TCP	60		51350 → 44818 [ACK] Seq=53 Ack=30 Win=29200 Len=0
▶ Frame 22: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0							
▶ Ethernet II, Src: Rockwell_a1:28:4c (00:1d:9c:a1:28:4c), Dst: Vmware_56:20:17 (00:0c:29:56:20:17)							
▶ Internet Protocol Version 4, Src: 192.168.0.59, Dst: 192.168.0.62							
▼ Transmission Control Protocol, Src Port: 44818, Dst Port: 51350, Seq: 29, Ack: 53, Len: 0							
Source Port: 44818							
Destination Port: 51350							
[Stream index: 1]							
[TCP Segment Len: 0]							
Sequence number: 29 (relative sequence number)							
Acknowledgment number: 53 (relative ack number)							
Header Length: 20 bytes							
▶ Flags: 0x011 [FIN, ACK]							
Window size value: 2000							
[Calculated window size: 2000]							
[Window size scaling factor: -2 (no window scaling used)]							
Checksum: 0x582b [unverified]							
[Checksum Status: Unverified]							
Urgent pointer: 0							
▶ [SEQ/ACK analysis]							
0000	00 0c 29 56 20 17 00 1d	9c a1 28 4c 00 00 45 00	..)V(L..E.				
0010	00 20 13 5d 00 00 00 06	a5 a9 c0 a0 00 3b c0 a0	.(.).....				
0020	00 3e af 12 c8 96 4b cc	23 c1 83 2d 63 aa 50 11	.>...K. #-.-c.P.				
0030	07 d0 58 2b 00 00 00 00	00 00 00 00 00 00 00 00	..X+....				

Figure 66. UnRegisterSession Response over TCP (Fuzzed Sender Context)

(4) T12 Results

No.	Time	Source	Destination	Protocol	Length	Resp	Info
12	10:52:43.212955	192.168.0.62	192.168.0.59	ENIP	82		Register Session (Req), Session: 0x00000000
13	10:52:43.226085	192.168.0.59	192.168.0.62	ENIP	82		Register Session (Rsp), Session: 0xA1764359
14	10:52:43.226699	192.168.0.62	192.168.0.59	TCP	60		48208 → 44818 [ACK] Seq=29 Ack=29 Win=29200 Len=0
15	10:52:43.316215	192.168.0.62	192.168.0.59	ENIP	78		Unregister Session (Req), Session: 0xA1764359
16	10:52:43.325985	192.168.0.59	192.168.0.62	ENIP	78		Unregister Session (Rsp), Session: 0xA1764359
▶ Frame 16: 78 bytes on wire (624 bits), 70 bytes captured (560 bits) on interface 0 ▶ Ethernet II, Src: Rockwell_a1:28:4c (00:1d:9c:a1:28:4c), Dst: Vmware_56:20:17 (00:0c:29:56:20:17) ▶ Internet Protocol Version 4, Src: 192.168.0.59, Dst: 192.168.0.62 ▶ Transmission Control Protocol, Src Port: 44818, Dst Port: 48208, Seq: 29, Ack: 53, Len: 24 ▼ EtherNet/IP (Industrial Protocol), Session: 0xA1764359, Unregister Session ▼ Encapsulation Header Command: Unregister Session (0x0066) Length: 0 Session Handle: 0xA1764359 Status: Incorrect Data (0x00000003) Sender Context: 0000000000000000 Options: 0x00000000							
0000	00 0c 29 56 20 17 00 1d	9c a1 28 4c 00 00 45 00	..JV(L..E.				
0010	00 40 00 fa 00 00 00 06	b7 f4 c0 a8 00 3b c0 a8	.@.....:.....				
0020	00 3e af 12 bc 50 49 3d	31 6f 4b f2 da 0c 50 18	..>...PI= 1oK...P.				
0030	07 d0 e1 26 00 00 66 00	00 00 59 43 76 a1 03 00	...&...f. ..YCV...				
0040	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00				

Figure 67. UnRegisterSession Response over TCP (Fuzzed Options)

No.	Time	Source	Destination	Protocol	Length	Resp	Info
1	13:06:55.466598	192.168.0.62	192.168.0.59	TCP	74		60700 → 44818 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=
2	13:06:55.467107	192.168.0.59	192.168.0.62	TCP	62		44818 → 60700 [SYN, ACK] Seq=0 Ack=1 Win=2000 Len=0 MSS=16384
3	13:06:55.467580	192.168.0.62	192.168.0.59	TCP	60		60700 → 44818 [ACK] Seq=1 Ack=1 Win=29200 Len=0
4	13:06:55.468378	192.168.0.62	192.168.0.59	ENIP	82		Register Session (Req), Session: 0x00000000
5	13:06:55.477481	192.168.0.59	192.168.0.62	ENIP	82		Register Session (Rsp), Session: 0x6AEE6AED
6	13:06:55.480986	192.168.0.62	192.168.0.59	TCP	60		60700 → 44818 [ACK] Seq=29 Ack=29 Win=29200 Len=0
7	13:06:55.583672	192.168.0.62	192.168.0.59	ENIP	78		Unregister Session (Req), Session: 0x6AEE6AED
8	13:06:55.586412	192.168.0.59	192.168.0.62	ENIP	78		Unregister Session (Rsp), Session: 0x6AEE6AED
9	13:06:55.586856	192.168.0.62	192.168.0.59	TCP	60		60700 → 44818 [ACK] Seq=53 Ack=53 Win=29200 Len=0
10	13:06:55.691892	192.168.0.62	192.168.0.59	CIP CM	142		Connection Manager - Forward Open (Message Router)
11	13:06:55.696682	192.168.0.59	192.168.0.62	CIP CM	124		Success: Connection Manager - Forward Open
12	13:06:55.696996	192.168.0.59	192.168.0.62	TCP	60		60700 → 44818 [ACK] Seq=141 Ack=123 Win=29200 Len=0
▶ Frame 11: 124 bytes on wire (992 bits), 124 bytes captured (992 bits) on interface 0 ▶ Ethernet II, Src: Rockwell_a1:28:4c (00:1d:9c:a1:28:4c), Dst: Vmware_56:20:17 (00:0c:29:56:20:17) ▶ Internet Protocol Version 4, Src: 192.168.0.59, Dst: 192.168.0.62 ▶ Transmission Control Protocol, Src Port: 44818, Dst Port: 60700, Seq: 53, Ack: 141, Len: 70 ▶ EtherNet/IP (Industrial Protocol), Session: 0x6AEE6AED, Send RR Data ▶ Common Industrial Protocol ▼ CIP Connection Manager Service: Forward Open (Response) Command Specific Data							
0000	00 0c 29 56 20 17 00 1d	9c a1 28 4c 00 00 45 00	..JV(L..E.				
0010	00 6e 01 3b 00 00 00 06	b7 85 c0 a8 00 3b c0 a8	.n;.....:.....				
0020	00 3e af 12 ed 1c 49 3a	a0 8a e0 1b fb 64 50 18	..>...I:.....dP.				
0030	07 d0 87 3e 00 00 6f 00	2e 00 ed 6a ee 6a 00 00	...>..o.j.j..				
0040	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00				
0050	00 00 00 04 02 00 00 00	00 00 b2 00 1e 00 d4 00				
0060	00 00 4b b8 6a ed 01 00	fe 00 02 00 4d 00 f3 0a	..K.j....M...				
0070	60 05 d0 12 13 00 d0 12	13 00 00 00				

Figure 68. CIP Forward Open Response Following ENIP UnRegisterSession Request with Fuzzed Options Field

(5) T13 Results

No.	Time	Source	Destination	Protocol	Length	Resp	Info
1	10:45:49.803150	Vmware_56:20:17	Broadcast	ARP	60		Who has 192.168.0.59? Tell 192.168.0.62
2	10:45:49.803334	Rockwell_a1:28:4c	Vmware_56:20:17	ARP	60		192.168.0.59 is at 00:1d:9c:a1:28:4c
3	10:45:49.805889	192.168.0.62	192.168.0.59	ENIP	66		Unregister Session (Req), Session: 0x00000000
4	10:45:49.807488	Rockwell_a1:28:4c	Broadcast	ARP	60		Who has 192.168.0.62? Tell 192.168.0.59
5	10:45:49.807750	Vmware_56:20:17	Rockwell_a1:28:4c	ARP	60		192.168.0.62 is at 00:0c:29:56:20:17
6	10:45:49.809282	192.168.0.59	192.168.0.62	ENIP	66		Unregister Session (Resp), Session: 0x00000000
7	10:45:49.810499	192.168.0.62	192.168.0.59	ICMP	94		Destination unreachable (Port unreachable)

▶ Frame 6: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
 ▶ Ethernet II, Src: Rockwell_a1:28:4c (00:1d:9c:a1:28:4c), Dst: Vmware_56:20:17 (00:0c:29:56:20:17)
 ▶ Internet Protocol Version 4, Src: 192.168.0.59, Dst: 192.168.0.62
 ▶ User Datagram Protocol, Src Port: 44818, Dst Port: 51606
 ▼ EtherNet/IP (Industrial Protocol), Session: 0x00000000, Unregister Session
 ▼ Encapsulation Header
 Command: Unregister Session (0x0066)
 Length: 0
 Session Handle: 0x00000000
 Status: Unknown (0x01000000)
 Sender Context: 0000000000000000
 Options: 0x00000000

```

0000 00 0c 29 56 20 17 0d 1d 9c a1 28 4c 00 00 45 00  ..V... (L..E.
0010 00 34 00 f7 00 00 00 11 b7 f8 c0 a0 00 3b c0 a8  .4.....
0020 00 3e af 12 c9 96 00 20 9f 39 66 00 00 00 00 00  .e.... .9f....
0030 00 00 00 00 01 00 00 00 00 00 00 00 00 00 00  ..
0040 00 00
  
```

Figure 69. UnRegisterSession Response over UDP (Functionality Test)

C. ENIP SENDRRDATA TEST CASES

This section shows the results of the ENIP SendRRData test cases.

(1) T14 Results

No.	Time	Source	Destination	Protocol	Length	Resp	Info
34	12:21:31.200785	192.168.0.62	192.168.0.59	ENIP	82		Register Session (Req), Session: 0x00000000
35	12:21:31.209666	192.168.0.59	192.168.0.62	ENIP	82		Register Session (Resp), Session: 0x60817FCE
36	12:21:31.213590	192.168.0.62	192.168.0.59	TCP	60		51390 → 44818 [ACK] Seq=29 Ack=29 Win=29200 Len=0
37	12:21:31.230250	192.168.0.62	192.168.0.59	CIP CM	140		Connection Manager - Forward Open (Message Router)
38	12:21:31.240927	192.168.0.59	192.168.0.62	CIP	124		Success: Service (0x54)
39	12:21:31.243499	192.168.0.62	192.168.0.59	TCP	60		51390 → 44818 [FIN, ACK] Seq=115 Ack=99 Win=29200 Len=0
40	12:21:31.245332	192.168.0.59	192.168.0.62	TCP	60		44818 → 51390 [ACK] Seq=99 Ack=116 Win=2000 Len=0

▶ Frame 37: 140 bytes on wire (1120 bits), 140 bytes captured (1120 bits) on interface 0
 ▶ Ethernet II, Src: Vmware_56:20:17 (00:0c:29:56:20:17), Dst: Rockwell_a1:28:4c (00:1d:9c:a1:28:4c)
 ▶ Internet Protocol Version 4, Src: 192.168.0.62, Dst: 192.168.0.59
 ▶ Transmission Control Protocol, Src Port: 51390, Dst Port: 44818, Seq: 29, Ack: 29, Len: 86
 ▼ EtherNet/IP (Industrial Protocol), Session: 0x00000001, Send RR Data
 ▼ Encapsulation Header
 Command: Send RR Data (0x006f)
 Length: 62
 Session Handle: 0x00000001
 Status: Success (0x00000000)
 Sender Context: 0000000000000000
 Options: 0x00000000
 ▶ Command Specific Data
 ▶ Common Industrial Protocol
 ▶ CIP Connection Manager

```

0000 00 1d 9c a1 28 4c 00 0c 29 56 20 17 00 00 45 00  ....(L..)V...E.
0010 00 7e cc e2 40 00 00 06 eb cd c0 a0 00 3e c0 a8  .~.Q..@.....
0020 00 3b c8 be af 12 c1 27 01 67 4b d7 04 fe 50 18  .;.... .gK...P.
0030 72 10 86 fd 00 00 6f 00 3e 00 01 00 00 00 00 00  r....o. >.....
0040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
0050 00 00 00 00 02 00 00 00 00 00 b2 00 2e 00 54 02  .....T.
0060 20 06 24 01 07 9b 02 00 00 00 01 00 fe 80 02 00  .$. ....
0070 44 00 f3 0a 60 05 02 00 02 00 d0 12 13 00 12 43  M... ..C
0080 d0 12 13 00 12 43 a3 02 20 02 24 01  .....C...$.
  
```

Figure 70. SendRRData Request over TCP (Fuzzed Session Handle)

No.	Time	Source	Destination	Protocol	Length	Resp	Info
34	12:21:31.200785	192.168.0.62	192.168.0.59	ENIP	82		Register Session (Req), Session: 0x00000000
35	12:21:31.209666	192.168.0.59	192.168.0.62	ENIP	82		Register Session (Rsp), Session: 0x60817fCE
36	12:21:31.213590	192.168.0.62	192.168.0.59	TCP	60		51390 → 44818 [ACK] Seq=29 Ack=29 Win=29200 Len=0
37	12:21:31.230250	192.168.0.62	192.168.0.59	CIP CM	140		Connection Manager – Forward Open (Message Router)
38	12:21:31.240927	192.168.0.59	192.168.0.62	CIP	124		Success: Service (0x54)
39	12:21:31.243499	192.168.0.62	192.168.0.59	TCP	60		51390 → 44818 [FIN, ACK] Seq=115 Ack=99 Win=29200 Len=0
40	12:21:31.245332	192.168.0.59	192.168.0.62	TCP	60		44818 → 51390 [ACK] Seq=99 Ack=115 Win=2000 Len=0
▶ Frame 38: 124 bytes on wire (992 bits), 124 bytes captured (992 bits) on interface 0 ▶ Ethernet II, Src: Rockwell_a1:28:4c (00:1d:9c:a1:28:4c), Dst: Vmware_56:20:17 (00:0c:29:56:20:17) ▶ Internet Protocol Version 4, Src: 192.168.0.59, Dst: 192.168.0.62 ▶ Transmission Control Protocol, Src Port: 44818, Dst Port: 51390, Seq: 29, Ack: 115, Len: 70 ▼ EtherNet/IP (Industrial Protocol), Session: 0x60817fCE, Send RR Data ▼ Encapsulation Header Command: Send RR Data (0x006f) Length: 46 Session Handle: 0x6d817fce Status: Success (0x00000000) Sender Context: 0000000000000000 Options: 0x00000000 ▶ Command Specific Data ▶ Common Industrial Protocol ▶ CIP Class Generic 0000 00 0c 29 56 20 17 00 1d 9c a1 28 4c 00 00 45 00 ...J.V... (L..E. 0010 00 5e 14 60 00 00 00 05 a4 60 c8 a0 00 3b c0 a8 ...n.... 0020 00 3e af 12 2f a0 69 79 4d 67 18 f3 50 18 ...X.../.. lYmg..P. 0030 07 d0 a6 5a 00 00 6f 00 2e 00 ce 7f 81 6d 00 00 ...Z..o...<...q.. 0040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0050 00 00 00 04 02 00 00 00 00 b2 00 2c 00 54 02T. 0060 00 00 19 0d 7f ce 01 00 00 02 00 4d 00 f3 0a ...y.....M.. 0070 60 05 d0 12 13 00 d0 12 13 00 00 00 							

Figure 71. SendRRData Response over TCP (Fuzzed Session Handle)

(2) T15 Results

No.	Time	Source	Destination	Protocol	Length	Resp	Info
56	21:58:44.943847	192.168.0.62	192.168.0.59	ENIP	82		Register Session (Req), Session: 0x00000000
57	21:58:44.955123	192.168.0.59	192.168.0.62	ENIP	82		Register Session (Rsp), Session: 0x71f6f63C
58	21:58:44.957375	192.168.0.62	192.168.0.59	TCP	60		43864 → 44818 [ACK] Seq=29 Ack=29 Win=29200 Len=0
59	21:58:44.965327	192.168.0.62	192.168.0.59	CIP CM	140		Connection Manager – Forward Open (Message Router)
60	21:58:44.974064	192.168.0.59	192.168.0.62	CIP CM	124		Success: Connection Manager – Forward Open
61	21:58:44.976822	192.168.0.62	192.168.0.59	TCP	60		43864 → 44818 [FIN, ACK] Seq=115 Ack=99 Win=29200 Len=0
62	21:58:44.978818	192.168.0.59	192.168.0.62	TCP	60		44818 → 43864 [ACK] Seq=99 Ack=116 Win=2000 Len=0
▶ Frame 59: 140 bytes on wire (1120 bits), 140 bytes captured (1120 bits) on interface 0 ▶ Ethernet II, Src: Vmware_56:20:17 (00:0c:29:56:20:17), Dst: Rockwell_a1:28:4c (00:1d:9c:a1:28:4c) ▶ Internet Protocol Version 4, Src: 192.168.0.62, Dst: 192.168.0.59 ▶ Transmission Control Protocol, Src Port: 43864, Dst Port: 44818, Seq: 29, Ack: 29, Len: 86 ▼ EtherNet/IP (Industrial Protocol), Session: 0x71f6f63C, Send RR Data ▼ Encapsulation Header Command: Send RR Data (0x006f) Length: 62 Session Handle: 0x71f6f63c Status: Unknown (0x00001111) Sender Context: 0000000000000000 Options: 0x00000000 ▶ Command Specific Data ▶ Common Industrial Protocol ▶ CIP Connection Manager 0000 00 1d 9c a1 28 4c 00 0c 29 56 20 17 00 00 45 00(L..)V...E. 0010 00 7e 79 7d 40 00 40 06 3f 33 c0 a0 00 3e c0 a8 ...y)8.. 73... 0020 00 3b ab 58 af 12 2f a0 69 79 4d 67 18 f3 50 18 ...X.../.. lYmg..P. 0030 72 10 74 da 00 00 6f 00 3e 00 3c f6 71 11 11 ...t...o...<...q.. 0040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0050 00 00 00 00 02 00 00 00 00 b2 00 2c 00 54 02T. 0060 20 06 24 01 07 0b 02 00 00 00 01 00 fe 00 02 00 ...\$. 0070 4d 00 f3 0a 60 05 02 00 02 00 d0 12 13 00 12 43 M.....C 0080 d0 12 13 00 12 43 a3 02 20 02 24 01C...\$. 							

Figure 72. SendRRData Request over TCP (Fuzzed Status)

No.	Time	Source	Destination	Protocol	Length	Resp	Info
56	21:58:44.943847	192.168.0.62	192.168.0.59	ENIP	82		Register Session (Req), Session: 0x00000000
57	21:58:44.955123	192.168.0.59	192.168.0.62	ENIP	82		Register Session (Rsp), Session: 0x71f6f63c
58	21:58:44.957375	192.168.0.62	192.168.0.59	TCP	60		43864 → 44818 [ACK] Seq=29 Ack=29 Win=29200 Len=0
59	21:58:44.966327	192.168.0.62	192.168.0.59	CIP CM	140		Connection Manager - Forward Open (Message Router)
60	21:58:44.974864	192.168.0.59	192.168.0.62	CIP CM	124		Success: Connection Manager - Forward Open
61	21:58:44.976822	192.168.0.62	192.168.0.59	TCP	60		43864 → 44818 [FIN, ACK] Seq=115 Ack=99 Win=29200 Len=0
62	21:58:44.978818	192.168.0.59	192.168.0.62	TCP	60		44818 → 43864 [ACK] Seq=99 Ack=116 Win=2000 Len=0
▶ Frame 60: 124 bytes on wire (992 bits), 124 bytes captured (992 bits) on interface 0 ▶ Ethernet II, Src: Rockwell_a1:28:4c (00:1d:9c:a1:28:4c), Dst: Vmware_56:20:17 (00:0c:29:56:20:17) ▶ Internet Protocol Version 4, Src: 192.168.0.59, Dst: 192.168.0.62 ▶ Transmission Control Protocol, Src Port: 44818, Dst Port: 43864, Seq: 29, Ack: 115, Len: 70 ▶ EtherNet/IP (Industrial Protocol), Session: 0x71f6f63c, Send RR Data							
▼ Encapsulation Header Command: Send RR Data (0x006f) Length: 46 Session Handle: 0x71f6f63c Status: Success (0x00000000) Sender Context: 0000000000000000 Options: 0x00000000							
▶ Command Specific Data ▶ Common Industrial Protocol ▶ CIP Connection Manager							
0000	00 0c 29 56 20 17 00 1d	9c a1 28 4c 00 00 45 00	..)V(L..E.				
0010	00 0e 34 00 00 00 06 b2	8c c0 a0 00 3b c0 a8	.n.4.....				
0020	00 3e af 12 ab 58 4d 67	18 f3 2f a0 69 cf 50 18	.w...Xmg .../..I.P.				
0030	07 d0 70 68 00 00 6f 00	2e 00 3c f6 71 00 00	.ph..o...<..q..				
0040	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00				
0050	00 00 00 04 02 00 00 00	00 00 b2 00 1e 00 d4 00				
0060	00 00 86 ec f6 3c 01 00	fe 80 02 00 4d 00 f3 0a				
0070	60 05 d0 12 13 00 d0 12	13 00 00 00				

Figure 73. SendRRData Response over TCP (Fuzzed Status)

(3) T16 Results

No.	Time	Source	Destination	Protocol	Length	Resp	Info
30	14:49:01.932605	192.168.0.62	192.168.0.59	ENIP	82		Register Session (Req), Session: 0x00000000
31	14:49:01.936074	192.168.0.59	192.168.0.62	ENIP	82		Register Session (Rsp), Session: 0xc370c865
32	14:49:01.936561	192.168.0.62	192.168.0.59	TCP	60		51520 → 44818 [ACK] Seq=29 Ack=29 Win=29200 Len=0
33	14:49:01.950857	192.168.0.62	192.168.0.59	CIP CM	140		Connection Manager - Forward Open (Message Router)
34	14:49:01.956434	192.168.0.59	192.168.0.62	CIP CM	124		Success: Connection Manager - Forward Open
35	14:49:01.968640	192.168.0.62	192.168.0.59	TCP	60		51520 → 44818 [FIN, ACK] Seq=115 Ack=99 Win=29200 Len=0
▶ Ethernet II, Src: Rockwell_a1:28:4c (00:1d:9c:a1:28:4c), Dst: Vmware_56:20:17 (00:0c:29:56:20:17) ▶ Internet Protocol Version 4, Src: 192.168.0.59, Dst: 192.168.0.62 ▶ Transmission Control Protocol, Src Port: 44818, Dst Port: 51520, Seq: 29, Ack: 115, Len: 70 ▶ EtherNet/IP (Industrial Protocol), Session: 0xc370c865, Send RR Data							
▼ Encapsulation Header Command: Send RR Data (0x006f) Length: 46 Session Handle: 0xc370c865 Status: Success (0x00000000) Sender Context: 0000000000000001 Options: 0x00000000							
▼ Command Specific Data Interface Handle: CIP (0x00000000) Timeout: 1024 ▶ Item Count: 2 [Request In: 33] [Time: 0.005577000 seconds]							
▶ Common Industrial Protocol ▶ CIP Connection Manager							
0000	00 0c 29 56 20 17 00 1d	9c a1 28 4c 00 00 45 00	..)V(L..E.				
0010	00 0e 2d 97 00 00 00 06	8b 29 c0 a0 00 3b c0 a8	.n-.....)				
0020	00 3e af 12 c9 40 4b fd	6e c1 50 62 cf 9f 50 18	.w...@K. n.Xb..P.				
0030	07 d0 21 06 00 00 6f 00	2e 00 65 c0 7d c3 00 00				
0040	00 00 00 00 00 00 00 00	00 01 00 00 00 00 00 00				
0050	00 00 00 04 02 00 00 00	00 00 b2 00 1e 00 d4 00				
0060	00 00 52 a2 c8 65 01 00	fe 80 02 00 4d 00 f3 0a	.R..e.M...				
0070	60 05 d0 12 13 00 d0 12	13 00 00 00				

Figure 74. SendRRData Response over TCP (Fuzzed Sender Context)

(4) T17 Results

No.	Time	Source	Destination	Protocol	Length	Resp	Info
56	14:43:42.692842	192.168.0.59	192.168.0.62	ENIP	82		Register Session (Rsp), Session: 0xDBB4D164
57	14:43:42.697382	192.168.0.62	192.168.0.59	TCP	60		51512 → 44818 [ACK] Seq=29 Ack=29 Win=29200 Len=0
58	14:43:42.704419	192.168.0.62	192.168.0.59	CIP CM	140		Connection Manager - Forward Open (Message Router)
59	14:43:42.712969	192.168.0.59	192.168.0.62	CIP CM	124		Success: Connection Manager - Forward Open
60	14:43:42.715337	192.168.0.59	192.168.0.59	TCP	60		51512 → 44818 [FIN, ACK] Seq=115 Ack=99 Win=29200 Len=0
61	14:43:42.717128	192.168.0.59	192.168.0.62	TCP	60		44818 → 51512 [ACK] Seq=99 Ack=116 Win=2000 Len=0
62	14:43:42.717876	192.168.0.59	192.168.0.62	TCP	60		44818 → 51512 [FIN, ACK] Seq=99 Ack=116 Win=2000 Len=0
▶ Frame 58: 140 bytes on wire (1120 bits), 140 bytes captured (1120 bits) on interface 0 ▶ Ethernet II, Src: Vmware_56:20:17 (00:0c:29:56:20:17), Dst: Rockwell_a1:28:4c (00:1d:9c:a1:28:4c) ▶ Internet Protocol Version 4, Src: 192.168.0.62, Dst: 192.168.0.59 ▶ Transmission Control Protocol, Src Port: 51512, Dst Port: 44818, Seq: 29, Ack: 29, Len: 86 ▼ EtherNet/IP (Industrial Protocol), Session: 0xDBB4D164, Send RR Data Encapsulation Header Command: Send RR Data (0x006f) Length: 62 Session Handle: 0xdbb4d164 Status: Success (0x00000000) Sender Context: 0000000000000000 Options: 0xffffffff ▶ Command Specific Data ▶ Common Industrial Protocol ▶ CIP Connection Manager							
0000	00 1d 9c a1 28 4c 00 0c	29 56 20 17 00 00 45 00	...	(L..)V ...E.			
0010	00 7e 42 eb 40 00 00 06	75 c5 c0 a8 00 3e c0 a8	..B.@.0. u....P.				
0020	00 3b c9 38 af 12 57 6c	1a 33 4b fa d0 0c 50 18	..B..Wl .3K...P.				
0030	72 10 f3 93 00 00 0f 00	3e 00 64 d1 b4 d0 00 000. >.d....				
0040	00 00 00 00 00 00 00 00	00 00 ff ff ff 00 00				
0050	00 00 00 00 02 00 00 00	00 00 b2 00 2e 00 54 02T.				
0060	20 05 24 01 07 9b 02 00	00 00 01 00 fe 00 02 00	..\$.				
0070	4d 00 f3 0a 60 05 02 00	02 00 00 12 13 00 12 43	M.....C				
0080	d0 12 13 00 12 43 a3 02	20 02 24 01C..9.				

Figure 75. SendRRData Request over TCP (Fuzzed Options)

No.	Time	Source	Destination	Protocol	Length	Resp	Info
56	14:43:42.692842	192.168.0.59	192.168.0.62	ENIP	82		Register Session (Rsp), Session: 0xDBB4D164
57	14:43:42.697382	192.168.0.62	192.168.0.59	TCP	60		51512 → 44818 [ACK] Seq=29 Ack=29 Win=29200 Len=0
58	14:43:42.704419	192.168.0.62	192.168.0.59	CIP CM	140		Connection Manager - Forward Open (Message Router)
59	14:43:42.712969	192.168.0.59	192.168.0.62	CIP CM	124		Success: Connection Manager - Forward Open
60	14:43:42.715337	192.168.0.62	192.168.0.59	TCP	60		51512 → 44818 [FIN, ACK] Seq=115 Ack=99 Win=29200 Len=0
61	14:43:42.717128	192.168.0.59	192.168.0.62	TCP	60		44818 → 51512 [ACK] Seq=99 Ack=116 Win=2000 Len=0
62	14:43:42.717876	192.168.0.59	192.168.0.62	TCP	60		44818 → 51512 [FIN, ACK] Seq=99 Ack=116 Win=2000 Len=0
▶ Frame 59: 124 bytes on wire (992 bits), 124 bytes captured (992 bits) on interface 0 ▶ Ethernet II, Src: Rockwell_a1:28:4c (00:1d:9c:a1:28:4c), Dst: Vmware_56:20:17 (00:0c:29:56:20:17) ▶ Internet Protocol Version 4, Src: 192.168.0.59, Dst: 192.168.0.62 ▶ Transmission Control Protocol, Src Port: 44818, Dst Port: 51512, Seq: 29, Ack: 115, Len: 70 ▼ EtherNet/IP (Industrial Protocol), Session: 0xDBB4D164, Send RR Data Encapsulation Header Command: Send RR Data (0x006f) Length: 46 Session Handle: 0xdbb4d164 Status: Success (0x00000000) Sender Context: 0000000000000000 Options: 0x00000000 ▶ Command Specific Data ▶ Common Industrial Protocol ▶ CIP Connection Manager							
0000	00 0c 29 56 20 17 00 1d	9c a1 28 4c 00 00 45 00	..V(L..E.				
0010	00 5e 2d 7f 00 00 00 06	80 41 c0 a8 00 3b c0 a8	..n-....A....				
0020	00 3e af 12 c9 38 4b fa	d0 0c 57 6c 1a 09 50 18	..-..0K...Wl...P.				
0030	07 d0 3e 38 00 00 0f 00	2e 00 64 d1 b4 db 00 00	..B..0. ..d....				
0040	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00				
0050	00 00 00 04 02 00 00 00	00 00 b2 00 1e 00 d4 00				
0060	00 00 4b 9d d1 64 01 00	fe 00 02 00 4d 00 f3 0a	..K..d.M...				
0070	60 05 d0 12 13 00 d0 12	13 00 00 00				

Figure 76. SendRRData Response over TCP (Fuzzed Options)

(5) T18 Results

No.	Time	Source	Destination	Protocol	Length	Resp	Info
49	14:53:39.363629	192.168.0.62	192.168.0.59	ENIP	82		Register Session (Req), Session: 0x00000000
50	14:53:39.378858	192.168.0.59	192.168.0.62	ENIP	82		Register Session (Rsp), Session: 0x07E46B50
51	14:53:39.377221	192.168.0.62	192.168.0.59	TCP	60		51534 → 44818 [ACK] Seq=29 Ack=29 Win=29200 Len=0
52	14:53:39.386059	192.168.0.62	192.168.0.59	ENIP	140		Send RR Data (Req)
53	14:53:39.390551	192.168.0.59	192.168.0.62	ENIP	78		Send RR Data (Rsp)
54	14:53:39.392921	192.168.0.62	192.168.0.59	TCP	60		51534 → 44818 [FIN, ACK] Seq=115 Ack=53 Win=29200 Len=0

▶ Frame 53: 78 bytes on wire (624 bits), 78 bytes captured (624 bits) on interface 0
 ▶ Ethernet II, Src: Rockwell_La1:28:4c (08:1d:9c:a1:28:4c), Dst: Vmware_56:20:17 (00:0c:29:56:20:17)
 ▶ Internet Protocol Version 4, Src: 192.168.0.59, Dst: 192.168.0.62
 ▶ Transmission Control Protocol, Src Port: 44818, Dst Port: 51534, Seq: 29, Ack: 115, Len: 24
 ▶ EtherNet/IP (Industrial Protocol), Session: 0x07E46B50, Send RR Data
 ▼ Encapsulation Header
 Command: Send RR Data (0x006f)
 Length: 0
 Session Handle: 0xd7e46b50
 Status: Incorrect Data (0x00000003)
 Sender Context: 0000000000000000
 Options: 0x00000000

```

0000 00 0c 29 56 20 17 00 1d 9c a1 28 4c 08 00 45 00  ..)V ... ..(L..E.
0010 00 40 2d c1 00 00 00 06 0b 2d c0 a0 00 3b c0 a8  .@~..... ~-....:
0020 00 3e a7 12 c9 4e 50 4c fb e4 43 05 9b 9b 50 18  .P...NPPL..C...P.
0030 07 d0 db a3 00 00 6f 00 00 00 50 5b e4 d7 03 00  .....O...Pk....
0040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .
  
```

Figure 77. SendRRData Response over TCP (Fuzzed Interface Handle)

(6) T19 Results

No.	Time	Source	Destination	Protocol	Length	Resp	Info
19	14:55:39.679154	192.168.0.62	192.168.0.59	TCP	60		51540 → 44818 [ACK] Seq=29 Ack=29 Win=29200 Len=0
20	14:55:39.688117	192.168.0.62	192.168.0.59	CIP CM	140		Connection Manager - Forward Open (Message Router)
21	14:55:39.690163	192.168.0.59	192.168.0.62	CIP CM	124		Success: Connection Manager - Forward Open
22	14:55:39.780375	192.168.0.62	192.168.0.59	TCP	60		51540 → 44818 [FIN, ACK] Seq=115 Ack=99 Win=29200 Len=0
23	14:55:39.780007	192.168.0.59	192.168.0.62	TCP	60		44818 → 51540 [ACK] Seq=99 Ack=115 Win=2000 Len=0
24	14:55:39.781541	192.168.0.59	192.168.0.62	TCP	60		44818 → 51540 [FIN, ACK] Seq=99 Ack=115 Win=2000 Len=0
25	14:55:39.781748	192.168.0.62	192.168.0.59	TCP	60		51540 → 44818 [ACK] Seq=116 Ack=100 Win=29200 Len=0

Command: Send RR Data (0x006f)
 Length: 62
 Session Handle: 0x29fcd7a
 Status: Success (0x00000000)
 Sender Context: 0000000000000000
 Options: 0x00000000
 ▼ Command Specific Data
 Interface Handle: CIP (0x00000000)
 Timeout: 1
 ▼ Item Count: 2
 ▶ Type ID: Null Address Item (0x0000)
 ▶ Type ID: Unconnected Data Item (0x00b2)
 [Response In: 21]
 ▶ Common Industrial Protocol
 ▶ CIP Connection Manager

```

0000 00 1d 9c a1 28 4c 00 0c 29 56 20 17 00 00 45 00  ....(L..)V...E.
0010 00 7e 27 c4 40 00 00 06 90 ec c0 a0 00 3e c0 a8  .~'.@. ....>..
0020 00 3b c9 54 a7 12 6d 74 32 f9 4c 03 a4 6f 50 18  .;T..mt 2.L..oP.
0030 72 10 91 e6 00 00 6f 00 3e 00 7a da fc 29 00 00  r.....O...>Z...
0040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
0050 00 00 01 00 02 00 00 00 00 00 b2 00 2e 00 54 02  .....T.
0060 20 06 24 01 07 9b 02 00 00 00 01 00 fe 00 02 00  .$......
0070 4d 00 f3 0a 60 05 02 00 02 00 00 12 13 00 12 43  M.....C
0080 d0 12 13 00 12 43 a3 02 20 02 24 01  .....C...$.
  
```

Figure 78. SendRRData Request over TCP (Fuzzed Timeout)

No.	Time	Source	Destination	Protocol	Length	Resp	Info
17	14:55:39.665866	192.168.0.62	192.168.0.59	ENIP	82		Register Session (Req), Session: 0x00000000
18	14:55:39.677996	192.168.0.59	192.168.0.62	ENIP	82		Register Session (Rsp), Session: 0x29FCDA7A
19	14:55:39.679154	192.168.0.62	192.168.0.59	TCP	60		51540 → 44818 [ACK] Seq=29 Ack=29 Win=29200 Len=0
20	14:55:39.688117	192.168.0.62	192.168.0.59	CIP CM	140		Connection Manager → Forward Open (Message Router)
21	14:55:39.698163	192.168.0.59	192.168.0.62	CIP CM	124		Success: Connection Manager → Forward Open
22	14:55:39.708375	192.168.0.62	192.168.0.59	TCP	60		51540 → 44818 [FIN, ACK] Seq=115 Ack=99 Win=29200 Len=0
▶ Transmission Control Protocol, Src Port: 44818, Dst Port: 51540, Seq: 29, Ack: 115, Len: 70							
▼ Ethernet/IP (Industrial Protocol), Session: 0x29FCDA7A, Send RR Data							
▼ Encapsulation Header							
Command: Send RR Data (0x006f)							
Length: 46							
Session Handle: 0x29fcd7a							
Status: Success (0x00000000)							
Sender Context: 0000000000000000							
Options: 0x00000000							
▼ Command Specific Data							
Interface Handle: CIP (0x00000000)							
Timeout: 1024							
Item Count: 2							
[Request In: 20]							
[Time: 0.010046000 seconds]							
▶ Common Industrial Protocol							
▶ CIP Connection Manager							
0000	00 0c 29 56 20 17 00 1d	9c a1 28 4c 00 00 45 00	..)V(L..E.				
0010	00 0e 2d d3 00 00 00 06	8a ed c0 a0 00 3b c0 a0	.n~.....				
0020	00 3e af 12 c9 54 4c 03	a4 6f 6d 74 33 4f 50 18	..>...TL. .ont30P.				
0030	07 d0 65 22 00 00 6f 00	2e 00 7a da fc 29 00 00	..e"..o. .z.z)..				
0040	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00				
0050	00 00 00 04 02 00 00 00	00 00 b2 00 1e 00 d4 00				
0060	00 00 ba ef da 7a 01 00	fe 00 02 00 4d 00 f3 0aZ.M...				
0070	60 05 d0 12 13 00 d0 12	13 00 00 00				

Figure 79. SendRRData Response over TCP (Fuzzed Timeout)

D. ENIP SENDUNITDATA TEST CASES

This section shows the results of the ENIP SendUnitData test cases.

(1) T20 Results

No.	Time	Source	Destination	Protocol	Length	Resp	Info
44	15:48:16.675054	192.168.0.62	192.168.0.59	ENIP	82		Register Session (Req), Session: 0x00000000
45	15:48:16.686607	192.168.0.59	192.168.0.62	ENIP	82		Register Session (Rsp), Session: 0xE6912DEE
46	15:48:16.688710	192.168.0.62	192.168.0.59	TCP	60		51578 → 44818 [ACK] Seq=29 Ack=29 Win=29200 Len=0
47	15:48:16.698261	192.168.0.62	192.168.0.59	CIP CM	140		Connection Manager → Forward Open (Message Router)
48	15:48:16.706915	192.168.0.59	192.168.0.62	CIP CM	124		Success: Connection Manager → Forward Open
49	15:48:16.715987	192.168.0.62	192.168.0.59	CIP	106		Identity → Get Attributes All
50	15:48:16.726555	192.168.0.59	192.168.0.62	CIP	130		Success: Get Attributes All
▶ Frame 49: 106 bytes on wire (848 bits), 106 bytes captured (848 bits) on interface 0							
▶ Ethernet II, Src: Vmware_S6:20:17 (00:0c:29:56:20:17), Dst: Rockwell_a1:28:4c (00:1d:9c:a1:28:4c)							
▶ Internet Protocol Version 4, Src: 192.168.0.62, Dst: 192.168.0.59							
▶ Transmission Control Protocol, Src Port: 51578, Dst Port: 44818, Seq: 115, Ack: 99, Len: 52							
▼ Ethernet/IP (Industrial Protocol), Session: 0x00000001, Send Unit Data							
▼ Encapsulation Header							
Command: Send Unit Data (0x0070)							
Length: 28							
Session Handle: 0x00000001							
Status: Success (0x00000000)							
Sender Context: 0000000000000000							
Options: 0x00000000							
▶ Command Specific Data							
▶ Common Industrial Protocol							
0000	00 1d 9c a1 28 4c 00 0c	29 56 20 17 00 00 45 00(L.. JV ...E.				
0010	00 5c 23 fb 40 00 40 06	94 d7 c0 a0 00 3e c0 a0	..V#@.>..				
0020	00 3b c9 7a af 12 c0 c2	04 50 00 5a 40 5c 50 18	..z.....ZPZ@P.				
0030	72 10 d8 0f 00 00 70 00	1c 00 01 00 00 00 00 00P.....				
0040	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00				
0050	00 00 00 00 02 00 a1 00	04 00 93 59 2d ee b1 00Y-....				
0060	00 00 02 00 01 02 20 01	24 01S.				

Figure 80. SendUnitData Request over TCP (Fuzzed Session Handle)

No.	Time	Source	Destination	Protocol	Length	Resp	Info
44	15:48:16.675054	192.168.0.62	192.168.0.59	ENIP	82		Register Session (Req), Session: 0x00000000
45	15:48:16.686687	192.168.0.59	192.168.0.62	ENIP	82		Register Session (Rsp), Session: 0xE6912DEE
46	15:48:16.688710	192.168.0.62	192.168.0.59	TCP	60		51578 → 44818 [ACK] Seq=29 Ack=29 Win=29200 Len=0
47	15:48:16.690261	192.168.0.62	192.168.0.59	CIP CM	140		Connection Manager → Forward Open (Message Router)
48	15:48:16.786915	192.168.0.59	192.168.0.62	CIP CM	124		Success: Connection Manager → Forward Open
49	15:48:16.715987	192.168.0.62	192.168.0.59	CIP	106		Identity - Get Attributes All
50	15:48:16.726555	192.168.0.59	192.168.0.62	CIP	138		Success: Get Attributes All
▶ Frame 50: 138 bytes on wire (1104 bits), 138 bytes captured (1104 bits) on interface 0 ▶ Ethernet II, Src: Rockwell_a1:28:4c (00:1d:9c:a1:28:4c), Dst: Vmware_56:20:17 (00:0c:29:56:20:17) ▶ Internet Protocol Version 4, Src: 192.168.0.59, Dst: 192.168.0.62 ▶ Transmission Control Protocol, Src Port: 44818, Dst Port: 51578, Seq: 99, Ack: 167, Len: 84 ▼ EtherNet/IP (Industrial Protocol), Session: 0xE6912DEE, Send Unit Data ▼ Encapsulation Header Command: Send Unit Data (0x0070) Length: 60 Session Handle: 0xE6912DEE Status: Success (0x00000000) Sender Context: 0000000000000000 Options: 0x00000000 ▶ Command Specific Data ▶ Common Industrial Protocol							
0000	00 0c 29 56 20 17 00 1d	9c a1 28 4c 00 00 45 00	..)V ... (L..E.				
0010	00 7c 4d 00 00 00 00 06	6b 32 c0 a0 00 3b c0 a8	. M.... k2....				
0020	00 3e af 12 c9 7a 50 5a	40 5c e0 c2 04 8e 50 18	...aPZ @...P.				
0030	07 00 e2 43 00 00 70 00	3c 00 ee 2d 91 e6 00 00	...C..p. <...-...				
0040	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00				
0050	00 00 00 00 02 00 a1 00	04 00 01 00 00 00 b1 00				
0060	28 00 02 00 01 00 00 00	01 00 0c 00 b9 00 02 0e	(.....				
0070	64 00 4c 28 a1 9c 13 31	37 36 33 2d 4c 31 36 42	d/L/...1 763-L168				
0080	57 41 20 42 2f 31 34 2e	30 30	WA B/14. 00				

Figure 81. SendUnitData Response over TCP (Fuzzed Session Handle)

(2) T21 Results

No.	Time	Source	Destination	Protocol	Length	Resp	Info
82	10:33:14.659718	192.168.0.62	192.168.0.59	ENIP	82		Register Session (Req), Session: 0x00000000
83	10:33:14.665469	192.168.0.59	192.168.0.62	ENIP	82		Register Session (Rsp), Session: 0x476EA8C7
84	10:33:14.665887	192.168.0.62	192.168.0.59	TCP	60		51734 → 44818 [ACK] Seq=29 Ack=29 Win=29200 Len=0
85	10:33:14.680030	192.168.0.62	192.168.0.59	CIP CM	140		Connection Manager → Forward Open (Message Router)
86	10:33:14.685771	192.168.0.59	192.168.0.62	CIP CM	124		Success: Connection Manager → Forward Open
87	10:33:14.698844	192.168.0.62	192.168.0.59	CIP	106		Identity - Get Attributes All
88	10:33:14.705462	192.168.0.59	192.168.0.62	CIP	138		Success: Identity - Get Attributes All
▶ Frame 87: 106 bytes on wire (848 bits), 106 bytes captured (848 bits) on interface 0 ▶ Ethernet II, Src: Vmware_56:20:17 (00:0c:29:56:20:17), Dst: Rockwell_a1:28:4c (00:1d:9c:a1:28:4c) ▶ Internet Protocol Version 4, Src: 192.168.0.62, Dst: 192.168.0.59 ▶ Transmission Control Protocol, Src Port: 51734, Dst Port: 44818, Seq: 115, Ack: 99, Len: 52 ▼ EtherNet/IP (Industrial Protocol), Session: 0x476EA8C7, Send Unit Data ▼ Encapsulation Header Command: Send Unit Data (0x0070) Length: 28 Session Handle: 0x476ea8c7 Status: Unknown (0x0000ffff) Sender Context: 0000000000000000 Options: 0x00000000 ▶ Command Specific Data ▶ Common Industrial Protocol							
0000	00 1d 9c a1 28 4c 00 0c	29 56 20 17 00 00 45 00(L..)V ...E.				
0010	00 5c 94 b0 40 00 40 06	24 1a c0 a0 00 3e c0 a8	\..\@..@. \$.....				
0020	00 3b ca 16 af 12 ba 5d	2c 4f 4d 55 bf fe 50 10] OMU..P.				
0030	72 10 0a 00 00 00 70 00	1c 00 c7 a8 6e 47 ff ff	...'.p.nG..				
0040	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00				
0050	00 00 00 00 02 00 a1 00	04 00 32 76 a8 c7 b1 00 :2V....				
0060	00 00 02 00 01 02 20 01	24 01 \$.				

Figure 82. SendUnitData Request over TCP (Fuzzed Status: 0x0000FFFF)

No.	Time	Source	Destination	Protocol	Length	Resp	Info
82	10:33:14.659718	192.168.0.62	192.168.0.59	ENIP	82		Register Session (Req), Session: 0x00000000
83	10:33:14.665469	192.168.0.59	192.168.0.62	ENIP	82		Register Session (Rsp), Session: 0x476EA8C7
84	10:33:14.665807	192.168.0.62	192.168.0.59	TCP	60		51734 → 44818 [ACK] Seq=29 Ack=29 Win=29200 Len=0
85	10:33:14.680030	192.168.0.62	192.168.0.59	CIP CM	140		Connection Manager - Forward Open (Message Router)
86	10:33:14.685771	192.168.0.59	192.168.0.62	CIP CM	124		Success: Connection Manager - Forward Open
87	10:33:14.698844	192.168.0.62	192.168.0.59	CIP	106		Identity - Get Attributes All
88	10:33:14.705462	192.168.0.59	192.168.0.62	CIP	138		Success: Identity - Get Attributes All
▶ Frame 88: 138 bytes on wire (1104 bits), 138 bytes captured (1104 bits) on interface 0 ▶ Ethernet II, Src: Rockwell_a1:28:4c (00:1d:9c:a1:28:4c), Dst: Vmware_56:20:17 (00:0c:29:56:20:17) ▶ Internet Protocol Version 4, Src: 192.168.0.59, Dst: 192.168.0.62 ▶ Transmission Control Protocol, Src Port: 44818, Dst Port: 51734, Seq: 99, Ack: 167, Len: 84 ▼ EtherNet/IP (Industrial Protocol), Session: 0x476EA8C7, Send Unit Data ▼ Encapsulation Header Command: Send Unit Data (0x0070) Length: 60 Session Handle: 0x476ea8c7 Status: Success (0x00000000) Sender Context: 0000000000000000 Options: 0x00000000 ▶ Command Specific Data ▶ Common Industrial Protocol							
0000	00 0c 29 56 20 17 00 1d	9c a1 28 4c 00 00 45 00	..)V(L..E.				
0010	00 7c 00 28 00 00 00 06	b8 8a c0 a8 00 3b c0 a8	.l.(....				
0020	00 3e af 12 ca 16 4d 55	bf fe ba 5d 2c 83 50 18	.P...MU ...).P.				
0030	07 d0 ad 9e 00 00 70 00	3c 00 c7 a8 6e 47 00 00D. <...nG..				
0040	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00				
0050	00 00 00 00 02 00 a1 00	04 00 01 00 00 00 b1 00				
0060	28 00 02 00 81 00 00 00	01 00 0c 00 b9 00 02 0e	(.....				
0070	64 00 4c 28 a1 9c 13 31	37 36 33 2d 4c 31 36 42	d.L[...1 763-L168				
0080	57 41 20 42 2f 31 34 2e	30 30	WA B/14. 00				

Figure 83. SendUnitData Response over TCP (Fuzzed Status: 0x0000FFFF)

(3) T22 Results

No.	Time	Source	Destination	Protocol	Length	Resp	Info
29	10:38:53.409006	192.168.0.59	192.168.0.62	CIP CM	124		Success: Connection Manager - Forward Open
30	10:38:53.418956	192.168.0.62	192.168.0.59	CIP	106		Identity - Get Attributes All
31	10:38:53.429375	192.168.0.59	192.168.0.62	CIP	138		Success: Identity - Get Attributes All
32	10:38:53.431701	192.168.0.62	192.168.0.59	TCP	60		51746 → 44818 [FIN, ACK] Seq=167 Ack=183 Win=29200 Len=0
33	10:38:53.432567	192.168.0.59	192.168.0.62	TCP	60		44818 → 51746 [ACK] Seq=183 Ack=168 Win=2000 Len=0
34	10:38:53.433303	192.168.0.59	192.168.0.62	TCP	60		44818 → 51746 [FIN, ACK] Seq=183 Ack=168 Win=2000 Len=0
▼ EtherNet/IP (Industrial Protocol), Session: 0x8A2CF849, Send Unit Data ▼ Encapsulation Header Command: Send Unit Data (0x0070) Length: 60 Session Handle: 0x8a2cf849 Status: Success (0x00000000) Sender Context: 0000000000000001 Options: 0x00000000 ▶ Command Specific Data Interface Handle: CIP (0x00000000) Timeout: 0 Item Count: 2 [Request In: 30] [Time: 0.010419000 seconds] ▶ Common Industrial Protocol							
0000	00 0c 29 56 20 17 00 1d	9c a1 28 4c 00 00 45 00	..)V(L..E.				
0010	00 7c 00 52 00 00 00 06	b8 8a c0 a8 00 3b c0 a8	.l.R....				
0020	00 3e af 12 ca 22 4d 59	1a 6e 3c ac 4e 27 50 18	.P...MW .nc.N'P.				
0030	07 d0 6e 99 00 00 70 00	3c 00 49 f8 2c 8a 00 00	..n...p. <.I.....				
0040	00 00 00 00 00 00 00 00	00 01 00 00 00 00 00 00				
0050	00 00 00 00 02 00 a1 00	04 00 01 00 00 00 b1 00				
0060	28 00 02 00 81 00 00 00	01 00 0c 00 b9 00 02 0e	(.....				
0070	64 00 4c 28 a1 9c 13 31	37 36 33 2d 4c 31 36 42	d.L[...1 763-L168				
0080	57 41 20 42 2f 31 34 2e	30 30	WA B/14. 00				

Figure 84. SendUnitData Response over TCP (Fuzzed Sender Context)

(4) T23 Results

No.	Time	Source	Destination	Protocol	Length	Resp	Info
19	10:30:53.533890	192.168.0.62	192.168.0.59	ENIP	82		Register Session (Req), Session: 0x00000000
20	10:30:53.548121	192.168.0.59	192.168.0.62	ENIP	82		Register Session (Rsp), Session: 0x58c7461b
21	10:30:53.549445	192.168.0.62	192.168.0.59	TCP	60		51728 → 44818 [ACK] Seq=29 Ack=29 Win=29200 Len=0
22	10:30:53.565414	192.168.0.62	192.168.0.59	CIP CM	140		Connection Manager → Forward Open (Message Router)
23	10:30:53.579341	192.168.0.59	192.168.0.62	CIP CM	124		Success: Connection Manager → Forward Open
24	10:30:53.598127	192.168.0.62	192.168.0.59	CIP	186		Identity → Get Attributes All
25	10:30:53.598601	192.168.0.59	192.168.0.62	CIP	138		Success: Identity → Get Attributes All
Transmission Control Protocol, Src Port: 51728, Dst Port: 44818, Seq: 115, Ack: 99, Len: 52							
EtherNet/IP (Industrial Protocol), Session: 0x58c7461b, Send Unit Data							
Encapsulation Header							
Command: Send Unit Data (0x0070)							
Length: 28							
Session Handle: 0x58c7461b							
Status: Invalid Command (0x00000001)							
Sender Context: 0000000000000000							
Options: 0x00000000							
Command Specific Data							
Interface Handle: CIP (0x00000000)							
Timeout: 0							
Item Count: 2							
[Response In: 25]							
Common Industrial Protocol							
0000	00 1d 9c a1 28 4c 00 0c	29 56 20 17 00 00 45 00(L..)V ...E.				
0010	00 5c e0 d4 40 00 40 06	d7 fd c0 a0 00 3e c0 a0	.\.@.0.>..				
0020	00 3b ca 10 af 12 0d fb	d5 c4 49 0c 00 ed 50 18~I...P.				
0030	72 10 b9 63 00 00 70 00	1c 00 1b 46 c7 50 01 00	r..c..p. ...F.X..				
0040	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00				
0050	00 00 00 00 02 00 a1 00	04 00 7e bd 46 1b b1 00~.F....				
0060	00 00 02 00 01 02 20 01	24 01 \$.				

Figure 85. SendUnitData Request over TCP (Fuzzed Options)

No.	Time	Source	Destination	Protocol	Length	Resp	Info
19	10:30:53.533890	192.168.0.62	192.168.0.59	ENIP	82		Register Session (Req), Session: 0x00000000
20	10:30:53.548121	192.168.0.59	192.168.0.62	ENIP	82		Register Session (Rsp), Session: 0x58c7461b
21	10:30:53.549445	192.168.0.62	192.168.0.59	TCP	60		51728 → 44818 [ACK] Seq=29 Ack=29 Win=29200 Len=0
22	10:30:53.565414	192.168.0.62	192.168.0.59	CIP CM	140		Connection Manager → Forward Open (Message Router)
23	10:30:53.579341	192.168.0.59	192.168.0.62	CIP CM	124		Success: Connection Manager → Forward Open
24	10:30:53.598127	192.168.0.62	192.168.0.59	CIP	186		Identity → Get Attributes All
25	10:30:53.598601	192.168.0.59	192.168.0.62	CIP	138		Success: Identity → Get Attributes All
EtherNet/IP (Industrial Protocol), Session: 0x58c7461b, Send Unit Data							
Encapsulation Header							
Command: Send Unit Data (0x0070)							
Length: 60							
Session Handle: 0x58c7461b							
Status: Success (0x00000000)							
Sender Context: 0000000000000000							
Options: 0x00000000							
Command Specific Data							
Interface Handle: CIP (0x00000000)							
Timeout: 0							
Item Count: 2							
[Request In: 24]							
[Time: 0.008474000 seconds]							
Common Industrial Protocol							
0000	00 0c 29 56 20 17 00 1d	9c a1 28 4c 00 00 45 00	..)V(L..E.				
0010	00 7c 00 12 00 00 00 06	b0 a0 c0 a0 00 3b c0 a0>.....				
0020	00 3e af 12 ca 10 49 0c	00 ed 0d fb 05 f8 50 18~I...P.				
0030	07 00 47 3d 00 00 70 00	3c 00 1b 46 c7 50 00 00	..G...p. <..F.X..				
0040	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00				
0050	00 00 00 00 02 00 a1 00	04 00 01 00 00 00 b1 00				
0060	28 00 02 00 01 00 00 00	01 00 0c 00 09 02 0e	(.....				
0070	64 00 4c 20 a1 9c 13 31	37 36 33 2d 4c 31 36 42	d.L[...1 763-L168				
0080	57 41 20 42 2f 31 34 2e	30 30	WA B/14. 00				

Figure 86. SendUnitData Response over TCP (Fuzzed Options)

(5) T24 Results

No.	Time	Source	Destination	Protocol	Length	Resp	Info
46	10:58:00.069579	192.168.0.62	192.168.0.59	TCP	60		51780 → 44818 [ACK] Seq=29 Ack=29 Win=29200 Len=0
47	10:58:00.093341	192.168.0.62	192.168.0.59	CIP CM	140		Connection Manager - Forward Open (Message Router)
48	10:58:00.106878	192.168.0.59	192.168.0.62	CIP CM	124		Success: Connection Manager - Forward Open
49	10:58:00.119230	192.168.0.62	192.168.0.59	ENIP	106		Send Unit Data (Req), CONID: 0x6AF50621
50	10:58:00.126771	192.168.0.59	192.168.0.62	ENIP	78		Send Unit Data (Rsp)
51	10:58:00.128766	192.168.0.62	192.168.0.59	TCP	60		51780 → 44818 [FIN, ACK] Seq=167 Ack=123 Win=29200 Len=0
▶ Frame 50: 78 bytes on wire (624 bits), 78 bytes captured (624 bits) on interface 0 ▶ Ethernet II, Src: Rockwell_a1:28:4c (00:1d:9c:a1:28:4c), Dst: Vmware_56:20:17 (00:0c:29:56:20:17) ▶ Internet Protocol Version 4, Src: 192.168.0.59, Dst: 192.168.0.62 ▶ Transmission Control Protocol, Src Port: 44818, Dst Port: 51780, Seq: 99, Ack: 167, Len: 24 ▶ EtherNet/IP (Industrial Protocol), Session: 0xc17cf56a, Send Unit Data ▼ Encapsulation Header Command: Send Unit Data (0x0070) Length: 0 Session Handle: 0xc17cf56a Status: Incorrect Data (0x00000003) Sender Context: 0000000000000000 Options: 0x00000000							
0000	00 0c 29 56 20 17 00 1d	9c a1 28 4c 00 00 45 00	..)V(L..E.				
0010	00 40 00 c8 00 00 80 86	b8 26 c0 a8 00 3b c0 a8	.@.....&....;				
0020	00 3e af 12 ca 44 49 1c	51 a6 5d a7 9d 5b 50 18	..>...DI. Q]..[P.				
0030	07 d0 bc 46 00 00 70 00	00 00 5a f5 7c c1 03 00	...F..p. ...]. ...				
0040	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00				

Figure 87. SendUnitData Response over TCP (Fuzzed Interface Handle)

(6) T25 Results

No.	Time	Source	Destination	Protocol	Length	Resp	Info
21	11:04:00.005621	192.168.0.62	192.168.0.59	ENIP	82		Register Session (Req), Session: 0x00000000
22	11:04:00.009982	192.168.0.59	192.168.0.62	ENIP	82		Register Session (Rsp), Session: 0x9f93caa7
23	11:04:00.090277	192.168.0.62	192.168.0.59	TCP	60		51780 → 44818 [ACK] Seq=29 Ack=29 Win=29200 Len=0
24	11:04:00.107989	192.168.0.62	192.168.0.59	CIP CM	140		Connection Manager - Forward Open (Message Router)
25	11:04:00.120257	192.168.0.59	192.168.0.62	CIP CM	124		Success: Connection Manager - Forward Open
26	11:04:00.130389	192.168.0.62	192.168.0.59	CIP	106		Identity - Get Attributes All
27	11:04:00.141731	192.168.0.59	192.168.0.62	CIP	138		Success: Identity - Get Attributes All
▶ Transmission Control Protocol, Src Port: 51780, Dst Port: 44818, Seq: 115, Ack: 99, Len: 52 ▼ EtherNet/IP (Industrial Protocol), Session: 0x9f93caa7, Send Unit Data ▼ Encapsulation Header Command: Send Unit Data (0x0070) Length: 28 Session Handle: 0x9f93caa7 Status: Success (0x00000000) Sender Context: 0000000000000000 Options: 0x00000000 ▼ Command Specific Data Interface Handle: CIP (0x00000000) Timeout: 1 Item Count: 2 ▶ [Response In: 27]							
▶ Common Industrial Protocol							
0000	00 1d 9c a1 28 4c 00 0c	29 56 20 17 00 00 45 00(L..)V ...E.				
0010	00 5c 56 d1 40 00 40 06	62 01 c0 a8 00 3e c0 a8	.V.@.0. b....>..				
0020	00 3b ca 4c af 12 9a 7c	88 21 49 1f 19 c2 50 18	.;L... .I...P.				
0030	72 10 fc c3 00 00 70 00	1c 00 a7 ca 93 9f 00 00	F....P.				
0040	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00				
0050	00 00 01 00 02 00 a1 00	04 00 85 03 cb a7 b1 00				
0060	08 00 02 00 01 02 20 01	24 01 \$.				

Figure 88. SendUnitData Request over TCP (Fuzzed Timeout)

No.	Time	Source	Destination	Protocol	Length	Resp	Info
21	11:04:00.005621	192.168.0.62	192.168.0.59	ENIP	82		Register Session (Req), Session: 0x00000000
22	11:04:00.009902	192.168.0.59	192.168.0.62	ENIP	82		Register Session (Resp), Session: 0x9f93CAA7
23	11:04:00.090277	192.168.0.62	192.168.0.59	TCP	60		51788 → 44818 [ACK] Seq=29 Ack=29 Win=29200 Len=0
24	11:04:00.107909	192.168.0.62	192.168.0.59	CIP CM	140		Connection Manager - Forward Open (Message Router)
25	11:04:00.128257	192.168.0.59	192.168.0.62	CIP CM	124		Success: Connection Manager - Forward Open
26	11:04:00.130389	192.168.0.62	192.168.0.59	CIP	106		Identity - Get Attributes All
27	11:04:00.141731	192.168.0.59	192.168.0.62	CIP	130		Success: Identity - Get Attributes All

Ethernet/IP (Industrial Protocol), Session: 0x9f93CAA7, Send Unit Data
 Encapsulation Header
 Command: Send Unit Data (0x0070)
 Length: 60
 Session Handle: 0x9f93caa7
 Status: Success (0x00000000)
 Sender Context: 0000000000000000
 Options: 0x00000000
 Command Specific Data
 Interface Handle: CIP (0x00000000)
 Timeout: 0
 Item Count: 2
 Request In: 26
 [Time: 0.011342000 seconds]
 Common Industrial Protocol
 0000 00 0c 29 56 20 17 00 1d 9c a1 28 4c 00 00 45 00 ..)V(L..E.
 0010 00 7c 00 e4 00 00 00 06 b7 c0 a0 00 0b c0 a0 .j.....\$....
 0020 00 3e af 12 ca 4c 49 1f 19 c2 9a 7c 08 55 50 18 .>...ZM] %>...UP.
 0030 07 00 16 70 00 00 70 00 3c 00 a7 ca 93 9f 00 00*.g....
 0040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 0050 00 00 00 00 02 00 a1 00 04 00 01 00 00 01 00
 0060 28 00 02 00 01 00 00 00 01 00 0c 00 19 00 02 0e (.
 0070 64 00 4c 28 a1 9c 13 31 37 36 33 2d 4c 31 36 42 d.L{...1 763-L168
 0080 57 41 20 42 2f 31 34 2e 30 30 WA B/14. 00

Figure 89. SendUnitData Response over TCP (Fuzzed Timeout)

E. ENIP RESERVED FOR LEGACY USE TEST CASES

This section shows the results of the ENIP Reserved for Legacy Use test cases.

(1) T26 Results

No.	Time	Source	Destination	Protocol	Length	Resp	Info
439	12:20:24.984647	192.168.0.62	192.168.0.59	ENIP	78		Unknown Command (0x0001) (Req)
440	12:20:24.987992	192.168.0.59	192.168.0.62	ENIP	120		Unknown Command (0x0001) (Resp)
441	12:20:25.025732	192.168.0.62	192.168.0.59	TCP	60		51802 → 44818 [ACK] Seq=3509 Ack=3593 Win=29200 Len=0
442	12:20:25.086648	192.168.0.62	192.168.0.59	ENIP	78		Unknown Command (0x0069) (Req)
443	12:20:25.090897	192.168.0.59	192.168.0.62	ENIP	78		Unknown Command (0x0069) (Resp)
444	12:20:25.137903	192.168.0.62	192.168.0.59	TCP	60		51802 → 44818 [ACK] Seq=3533 Ack=3617 Win=29200 Len=0
445	12:20:25.188668	192.168.0.62	192.168.0.59	ENIP	78		Unknown Command (0x0002) (Req)

Frame 440: 120 bytes on wire (960 bits), 120 bytes captured (960 bits) on interface 0
 Ethernet II, Src: Rockwell_a1:28:4c (00:1d:9c:a1:28:4c), Dst: Vmware_56:20:17 (00:0c:29:56:20:17)
 Internet Protocol Version 4, Src: 192.168.0.59, Dst: 192.168.0.62
 Transmission Control Protocol, Src Port: 44818, Dst Port: 51802, Seq: 3527, Ack: 3509, Len: 66
 Ethernet/IP (Industrial Protocol), Session: 0x11f0671a, Unknown Command (0x0001)
 Encapsulation Header
 Command: Unknown (0x0001)
 Length: 42
 Session Handle: 0x11fd671a
 Status: Success (0x00000000)
 Sender Context: 0000000000000000
 Options: 0x00000000
 Encap Data: 010001002400010000000002af12c0a003b000000000000...
 Command Specific Data
 0000 00 0c 29 56 20 17 00 1d 9c a1 28 4c 00 00 45 00 ..)V(L..E.
 0010 00 6a 01 a0 00 00 00 06 b7 24 c0 a0 00 0b c0 a0 .j.....\$....
 0020 00 3e af 12 ca 5a 4d 6a 25 26 0d 13 0b 55 50 18 .>...ZM] %>...UP.
 0030 07 00 15 e6 00 00 01 00 2a 00 1a 67 fd 11 00 00*.g....
 0040 00 00 00 00 00 00 00 00 00 00 00 00 00 01 00
 0050 01 00 24 00 01 00 00 00 00 02 af 12 c0 a0 00 3b ..\$.
 0060 00 00 00 00 00 00 00 00 31 39 32 2e 31 36 38 2e 192.168.
 0070 30 2e 35 39 00 00 00 00 00 00 00 00 00 00 00 00 0.59....

Figure 90. Reserved for Legacy Use Response over TCP (Fuzzed Command Field)

(2) T27 Results

No.	Time	Source	Destination	Protocol	Length	Resp	Info
1234	12:25:41.671563	192.168.0.62	192.168.0.59	ENIP	66	Cancel	(Req)
1235	12:25:41.673628	192.168.0.59	192.168.0.62	ENIP	66	Cancel	(Rsp)
1236	12:25:41.682472	192.168.0.62	192.168.0.59	ENIP	66	Unknown Command	(0x006a) (Req)
1237	12:25:41.683623	192.168.0.59	192.168.0.62	ENIP	66	Unknown Command	(0x006a) (Rsp)
1238	12:25:41.693682	192.168.0.62	192.168.0.59	ENIP	66	Unknown Command	(0x008e) (Req)
1239	12:25:41.695648	192.168.0.59	192.168.0.62	ENIP	66	Unknown Command	(0x008e) (Rsp)
1240	12:25:41.702496	192.168.0.62	192.168.0.59	ENIP	66	Unknown Command	(0x0092) (Req)
▶ Frame 1235: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0							
▶ Ethernet II, Src: Rockwell_a1:28:4c (00:1d:9c:a1:28:4c), Dst: Vmware_56:20:17 (00:0c:29:56:20:17)							
▶ Internet Protocol Version 4, Src: 192.168.0.59, Dst: 192.168.0.62							
▶ User Datagram Protocol, Src Port: 44818, Dst Port: 51606							
▼ EtherNet/IP (Industrial Protocol), Session: 0x00000000, Cancel							
▼ Encapsulation Header							
Command: Cancel (0x0073)							
Length: 0							
Session Handle: 0x00000000							
Status: Unknown (0x01000000)							
Sender Context: 0000000000000000							
Options: 0x00000000							
0000	00 0c 29 56 20 17 00 1d	9c a1 28 4c 00 00 45 00	..)V(L..E.				
0010	00 34 09 cc 00 00 00 11	af 23 c0 a0 00 3b c0 a0	.4.....#.....				
0020	00 3e af 12 c9 96 00 20	92 39 73 00 00 00 00 00	.9.....35.....				
0030	00 00 00 00 00 01 00 00	00 00 00 00 00 00 00 00				
0040	00 00		..				

Figure 91. Reserved for Legacy Use Response over UDP
(Fuzzed Command Field)

F. ENIP RESERVED FOR FUTURE USE TEST CASES

This section shows the results of the ENIP Reserved for Future Use test cases.

(1) T28 Results

No.	Time	Source	Destination	Protocol	Length	Resp	Info
9	12:33:18.961816	192.168.0.62	192.168.0.59	ENIP	78	Unknown Command	(0x7398) (Req)
10	12:33:18.971295	192.168.0.59	192.168.0.62	ENIP	78	Unknown Command	(0x7398) (Rsp)
11	12:33:19.012865	192.168.0.62	192.168.0.59	TCP	60	51808 → 44818 [ACK] Seq=53 Ack=53 Win=29200 Len=0	
12	12:33:19.067169	192.168.0.62	192.168.0.59	ENIP	78	Unknown Command	(0x8255) (Req)
13	12:33:19.071162	192.168.0.59	192.168.0.62	ENIP	78	Unknown Command	(0x8255) (Rsp)
14	12:33:19.071601	192.168.0.62	192.168.0.59	TCP	60	51808 → 44818 [ACK] Seq=77 Ack=77 Win=29200 Len=0	
15	12:33:19.170390	192.168.0.62	192.168.0.59	ENIP	78	Unknown Command	(0x4c6a) (Req)
▶ Frame 13: 78 bytes on wire (624 bits), 78 bytes captured (624 bits) on interface 0							
▶ Ethernet II, Src: Rockwell_a1:28:4c (00:1d:9c:a1:28:4c), Dst: Vmware_56:20:17 (00:0c:29:56:20:17)							
▶ Internet Protocol Version 4, Src: 192.168.0.59, Dst: 192.168.0.62							
▶ Transmission Control Protocol, Src Port: 44818, Dst Port: 51808, Seq: 53, Ack: 77, Len: 24							
▼ EtherNet/IP (Industrial Protocol), Session: 0xc108a289, Unknown Command (0x8255)							
▼ Encapsulation Header							
Command: Unknown (0x8255)							
Length: 0							
Session Handle: 0xc108a289							
Status: Incorrect Data (0x00000003)							
Sender Context: 0000000000000000							
Options: 0x00000000							
0000	00 0c 29 56 20 17 00 1d	9c a1 28 4c 00 00 45 00	..)V(L..E.				
0010	00 40 15 52 00 00 00 06	a3 9c c0 a0 00 3b c0 a0	.0.R.....				
0020	00 3e af 12 ca 60 49 22	c9 33 7c 4c 83 31 50 18	.9...I" .3[L.I.P.				
0030	07 00 ff ec 00 00 55 02	00 00 09 a2 b8 c1 03 00U.				
0040	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00				

Figure 92. Reserved for Future Use Response over TCP
(Fuzzed Command Field)

(2) T29 Results

No.	Time	Source	Destination	Protocol	Length	Resp	Info
8	13:00:26.199983	192.168.0.62	192.168.0.59	ENIP	66		Unknown Command (0x0202) (Req)
9	13:00:26.201672	192.168.0.59	192.168.0.62	ENIP	66		Unknown Command (0x0202) (Rsp)
10	13:00:26.202150	192.168.0.62	192.168.0.59	ICMP	94		Destination unreachable (Port unreachable)
11	13:00:26.236361	192.168.0.62	192.168.0.59	ENIP	66		Unknown Command (0xb4b9) (Req)
12	13:00:26.237626	192.168.0.59	192.168.0.62	ENIP	66		Unknown Command (0xb4b9) (Rsp)
13	13:00:26.239240	192.168.0.62	192.168.0.59	ICMP	94		Destination unreachable (Port unreachable)
14	13:00:26.269180	192.168.0.62	192.168.0.59	ENIP	66		Unknown Command (0x6301) (Req)
▶ Frame 9: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0 ▶ Ethernet II, Src: Rockwell_L_a1:28:4c (00:1d:9c:a1:28:4c), Dst: Vmware_56:20:17 (00:0c:29:56:20:17) ▶ Internet Protocol Version 4, Src: 192.168.0.59, Dst: 192.168.0.62 ▶ User Datagram Protocol, Src Port: 44818, Dst Port: 51606 ▼ EtherNet/IP (Industrial Protocol), Session: 0x00000000, Unknown Command (0x0202) ▼ Encapsulation Header Command: Unknown (0x0202) Length: 0 Session Handle: 0x00000000 Status: Unknown (0x01000000) Sender Context: 0000000000000000 Options: 0x00000000							
0000	00 0c 29 56 20 17 00 1d	9c a1 28 4c 00 00 45 00	..)V(L..E.				
0010	00 34 00 02 00 00 00 11	b0 ed c0 a0 00 3b c0 a0	.4.....;..				
0020	00 3e af 12 c9 96 00 20	03 38 02 02 00 00 00 00	.>.....8.....				
0030	00 00 00 00 00 01 00 00	00 00 00 00 00 00 00 00				
0040	00 00		..				

Figure 93. Reserved for Future Use Response over UDP
(Fuzzed Command Field)

APPENDIX B. CIP COMMAND RESPONSES

The following Wireshark captures in Figures 94–118 illustrate test case responses for each command. For certain test cases, the corresponding request command sent to the SUT is also included to show how select fuzzed field inputs affect SUT responses. For descriptions of SUT responses, see Chapter V: Test Analysis.

A. CIP GET_ATTRIBUTES_ALL TEST CASES

This section shows the results of the CIP Get_Attributes_All test cases.

(1) T30 Results

The Get_Attributes_All request with a fuzzed Class field returns three types of responses. Figure 94 illustrates a successful CIP response. Figure 95 shows a “Service not supported” response. Figure 96 depicts a “Path destination unknown” response.

No.	Time	Source	Destination	Protocol	Length	Resp	Info
4	11:00:55.522701	192.168.0.62	192.168.0.59	ENIP	82		Register Session (Req), Session: 0x00000000
5	11:00:55.528696	192.168.0.59	192.168.0.62	ENIP	82		Register Session (Rsp), Session: 0x44E6647D
6	11:00:55.528999	192.168.0.62	192.168.0.59	TCP	60		49234 → 44818 [ACK] Seq=29 Ack=29 Win=29200 Len=0
7	11:00:55.545100	192.168.0.62	192.168.0.59	CIP CM	140		Connection Manager – Forward Open (Message Router)
8	11:00:55.558528	192.168.0.59	192.168.0.62	CIP CM	124		Success: Connection Manager – Forward Open
9	11:00:55.568665	192.168.0.62	192.168.0.59	CIP	106		Identity – Get Attributes All
10	11:00:55.578369	192.168.0.59	192.168.0.62	CIP	138		Success: Identity – Get Attributes All

▶ Ethernet/IP (Industrial Protocol), Session: 0x44E6647D, Send Unit Data

▼ Common Industrial Protocol

- ▶ Service: Get Attributes All (Response)
- ▶ Status: Success:
 - [Request Path Size: 2 (words)]
 - [Request Path: Identity, Instance: 0x01]
- ▶ Get Attributes All (Response)
 - ▼ Attribute: 1 (Vendor ID)
 - Vendor ID: 1
 - ▼ Attribute: 2 (Device Type)
 - Device Type: 12
 - ▼ Attribute: 3 (Product Code)
 - Product Code: 185
 - ▼ Attribute: 4 (Revision)
 - Major Revision: 2
 - Minor Revision: 14
 - ▼ Attribute: 5 (Status)
 - Status: 0x0064
 - ▼ Attribute: 6 (Serial Number)
 - Serial Number: 0x9ca1284c
 - ▼ Attribute: 7 (Product Name)
 - Product Name: 1763-L168WA B/14.00

0000

00 0c 29 56 20 17 00 1d 9c a1 28 4c 08 00 45 00 ..)V(L..E.

0010

00 7c 01 02 00 00 00 06 b7 b0 c0 a8 00 3b c0 a8 .|.

0020

00 3e 0f 12 9d 2a 49 0d d3 07 70 19 5e 63 50 18 .>...*I. . . . ^CP.

0030

07 00 06 74 00 00 70 00 3c 00 7d 64 e6 44 00 00 ...t..p. <.)d.O..

0040

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

0050

00 00 00 00 02 00 a1 00 04 00 01 00 00 00 b1 00

0060

28 00 02 00 01 00 00 00 01 00 0c 00 b9 00 02 0e (.

0070

64 00 4c 28 a1 9c 13 31 37 36 33 2d 4c 31 36 42 d.L(. ...1 763-L168

0080

57 41 20 42 2f 31 34 2e 30 30 WA B/14. 00

Figure 94. Get_Attributes_All Response over TCP (Class 0x01, Instance 0x01)

No.	Time	Source	Destination	Protocol	Length	Resp	Info
858	11:02:21.996664	192.168.0.59	192.168.0.62	CIP	104		Path destination unknown: Class (0x8e) - Get Attributes All
859	11:02:22.000615	192.168.0.62	192.168.0.59	CIP	106		Base Switch - Get Attributes All
860	11:02:22.007797	192.168.0.59	192.168.0.62	CIP	104		Path destination unknown: Base Switch - Get Attributes All
861	11:02:22.013836	192.168.0.62	192.168.0.59	CIP	106		Connection Manager - Get Attributes All
862	11:02:22.027085	192.168.0.59	192.168.0.62	CIP	104		Service not supported: Connection Manager - Get Attributes All
▶ Frame 862: 104 bytes on wire (832 bits), 104 bytes captured (832 bits) on interface 0 ▶ Ethernet II, Src: Rockwell_a1:28:4c (00:1d:9c:a1:28:4c), Dst: Vmware_56:20:17 (00:0c:29:56:20:17) ▶ Internet Protocol Version 4, Src: 192.168.0.59, Dst: 192.168.0.62 ▶ Transmission Control Protocol, Src Port: 44818, Dst Port: 40238, Seq: 21567, Ack: 22319, Len: 50 ▶ EtherNet/IP (Industrial Protocol), Session: 0x8BB8FCFE, Send Unit Data ▼ Common Industrial Protocol ▼ Service: Get Attributes All (Response) 1... = Request/Response: Response (0x1) .000 0001 = Service: Get Attributes All (0x01) ▼ Status: Service not supported: General Status: Service not supported (0x08) Additional Status Size: 0 (words) [Request Path Size: 2 (words)] [Request Path: Connection Manager, Instance: 0x01] ▼ [Path Segment: 0x20 (8-Bit Class Segment)] [001. = Path Segment Type: Logical Segment (1)] [...0 00.. = Logical Segment Type: Class ID (0)] [.... ..00 = Logical Segment Format: 8-bit Logical Segment (0)] ▼ [8-Bit Class Segment] [Class: Connection Manager (0x06)] ▼ [Path Segment: 0x24 (8-Bit Instance Segment)] [001. = Path Segment Type: Logical Segment (1)] [...0 01.. = Logical Segment Type: Instance ID (1)] [.... ..00 = Logical Segment Format: 8-bit Logical Segment (0)] ▼ [8-Bit Instance Segment] [Instance: 0x01] 0000 00 0c 29 56 20 17 00 1d 9c a1 28 4c 00 00 45 00 ..)V(L..E. 0010 00 5a 06 ef 00 00 00 06 b1 e5 c0 a8 00 3b c0 a8 .Z..... 0020 00 3e af 12 9d 2e 49 0f 7f 6f f1 81 50 bc 50 18 .>....I. .o..P.P. 0030 07 d0 22 a6 00 00 70 00 1a 00 fe cc 8f 8b 00 00 .."....P. 0040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0050 00 00 00 00 02 00 a1 00 04 00 01 00 00 00 b1 00 0060 06 00 ac 01 81 00 00 00							

Figure 95. Get_Attributes_All “Service Not Supported” Response over TCP
(Class 0x06, Instance 0x01)

No.	Time	Source	Destination	Protocol	Length	Resp	Info
20	11:02:16.216778	192.168.0.59	192.168.0.62	CIP	104		Path destination unknown: Class (0x6a) - Get Attributes All
21	11:02:16.221339	192.168.0.62	192.168.0.59	CIP	106		Motor Data - Get Attributes All
22	11:02:16.226846	192.168.0.59	192.168.0.62	CIP	104		Path destination unknown: Motor Data - Get Attributes All
23	11:02:16.232981	192.168.0.62	192.168.0.59	CIP	106		Class (0x70) - Get Attributes All
24	11:02:16.246960	192.168.0.59	192.168.0.62	CIP	104		Path destination unknown: Class (0x70) - Get Attributes All
▶ EtherNet/IP (Industrial Protocol), Session: 0x8BB8FCFE, Send Unit Data ▼ Common Industrial Protocol ▼ Service: Get Attributes All (Response) 1... = Request/Response: Response (0x1) .000 0001 = Service: Get Attributes All (0x01) ▼ Status: Path destination unknown: General Status: Path destination unknown (0x05) Additional Status Size: 0 (words) [Request Path Size: 2 (words)] [Request Path: Motor Data, Instance: 0x01] ▼ [Path Segment: 0x20 (8-Bit Class Segment)] [001. = Path Segment Type: Logical Segment (1)] [...0 00.. = Logical Segment Type: Class ID (0)] [.... ..00 = Logical Segment Format: 8-bit Logical Segment (0)] ▼ [8-Bit Class Segment] [Class: Motor Data (0x28)] ▼ [Path Segment: 0x24 (8-Bit Instance Segment)] [001. = Path Segment Type: Logical Segment (1)] [...0 01.. = Logical Segment Type: Instance ID (1)] [.... ..00 = Logical Segment Format: 8-bit Logical Segment (0)] ▼ [8-Bit Instance Segment] [Instance: 0x01] 0000 00 0c 29 56 20 17 00 1d 9c a1 28 4c 00 00 45 00 ..)V(L..E. 0010 00 5a 05 4b 00 00 00 06 b3 89 c0 a8 00 3b c0 a8 .Z.K..... 0020 00 3e af 12 9d 2e 49 0f 2c bf f1 80 fb 6c 50 18 .>....I.IP. 0030 07 d0 71 a8 00 00 70 00 1a 00 fe cc 8f 8b 00 00 ..q....P. 0040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0050 00 00 00 00 02 00 a1 00 04 00 01 00 00 00 b1 00 0060 06 00 00 00 81 00 05 00							

Figure 96. Get_Attributes_All “Path Destination Unknown” Response over TCP
(Class 0x28, Instance 0x01)

(2) T31 Results

Figure 97 illustrates a “Path destination unknown response. Figure 98 demonstrates the response for the Identity Class with Instance 0x00.

No.	Time	Source	Destination	Protocol	Length	Resp	Info
14	11:04:25.305272	192.168.0.59	192.168.0.62	CIP CM	124		Success: Connection Manager – Forward Open
15	11:04:25.311826	192.168.0.62	192.168.0.59	CIP	106		Identity – Get Attributes All
16	11:04:25.324355	192.168.0.59	192.168.0.62	CIP	104		Path destination unknown: Identity – Get Attributes All
17	11:04:25.328677	192.168.0.62	192.168.0.59	CIP	106		Identity – Get Attributes All
18	11:04:25.334278	192.168.0.59	192.168.0.62	CIP	104		Path destination unknown: Identity – Get Attributes All

► Ethernet/IP (Industrial Protocol), Session: 0x7B33BEF0, Send Unit Data

▼ Common Industrial Protocol

▼ Service: Get Attributes All (Response)

1... = Request/Response: Response (0x1)

.000 0001 = Service: Get Attributes All (0x01)

▼ Status: Path destination unknown:

General Status: Path destination unknown (0x05)

Additional Status Size: 0 (words)

[Request Path Size: 2 (words)]

▼ [Request Path: Identity, Instance: 0x16]

▼ [Path Segment: 0x20 (8-Bit Class Segment)]

[001. = Path Segment Type: Logical Segment (1)]

[...0 00.. = Logical Segment Type: Class ID (0)]

[.... ..00 = Logical Segment Format: 8-bit Logical Segment (0)]

▼ [8-Bit Class Segment]

[Class: Identity (0x01)]

▼ [Path Segment: 0x24 (8-Bit Instance Segment)]

[001. = Path Segment Type: Logical Segment (1)]

[...0 01.. = Logical Segment Type: Instance ID (1)]

[.... ..00 = Logical Segment Format: 8-bit Logical Segment (0)]

▼ [8-Bit Instance Segment]

[Instance: 0x16]

0000 00 0c 29 56 20 17 00 1d 9c a1 28 4c 08 00 45 00 ..)V(L..E.

0010 00 5a 09 a4 00 00 00 06 af 30 c0 a8 00 3b c0 a8 .Z..... .0...;..

0020 00 3e af 12 9d 30 4d 57 c4 27 e3 71 f7 b4 50 18 .>...0MW .'.q..P.

0030 07 d8 57 db 00 00 70 00 1a 00 f0 be 33 7b 00 00 ..W...P.3{..

0040 00 00 00 00 00 00 00 00 00 00 00 00 00 00

0050 00 00 00 02 00 a1 00 04 00 01 00 00 00 b1 00

0060 06 00 02 00 81 00 05 00

Figure 97. Get_Attributes_All “Path Destination Unknown” Response over TCP (Class 0x01, Instance 0x16)

No.	Time	Source	Destination	Protocol	Length	Resp	Info
1723	11:04:35.918642	192.168.0.62	192.168.0.59	CIP	106		Identity – Get Attributes All
1724	11:04:35.924592	192.168.0.59	192.168.0.62	CIP	112		Success: Identity – Get Attributes All
1725	11:04:35.928966	192.168.0.62	192.168.0.59	CIP	106		Identity – Get Attributes All
1726	11:04:35.933949	192.168.0.59	192.168.0.62	CIP	104		Path destination unknown: Identity – Get Attributes All

► Transmission Control Protocol, Src Port: 44818, Dst Port: 48240, Seq: 42915, Ack: 44575, Len: 58

► Ethernet/IP (Industrial Protocol), Session: 0x7B33BEF0, Send Unit Data

▼ Common Industrial Protocol

▼ Service: Get Attributes All (Response)

1... = Request/Response: Response (0x1)

.000 0001 = Service: Get Attributes All (0x01)

▼ Status: Success:

General Status: Success (0x00)

Additional Status Size: 0 (words)

[Request Path Size: 2 (words)]

▼ [Request Path: Identity, Instance: 0x00]

► [Path Segment: 0x20 (8-Bit Class Segment)]

► [Path Segment: 0x24 (8-Bit Instance Segment)]

▼ Get Attributes All (Response)

▼ Attribute: 1 (Revision)

Revision: 1

▼ Attribute: 2 (Max Instance)

Max Instances: 1

▼ Attribute: 6 (Maximum ID Number Class Attributes)

Maximum ID Number Class Attributes: 0

▼ Attribute: 7 (Maximum ID Number Instance Attributes)

Maximum ID Number Instance Attributes: 0

0000 00 0c 29 56 20 17 00 1d 9c a1 28 4c 08 00 45 00 ..)V(L..E.

0010 00 52 0c fa 00 00 00 06 ab d2 c0 a8 00 3b c0 a8 .b..... ..

0020 00 3e af 12 9d 30 4d 58 6b 67 c3 72 a5 2c 50 18 .>...0MX kg.r..P.

0030 07 d8 a0 16 00 00 70 00 22 00 f0 be 33 7b 00 00p. "...3{..

0040 00 00 00 00 00 00 00 00 00 00 00 00 00 00

0050 00 00 00 02 00 a1 00 04 00 01 00 00 00 b1 00

0060 0e 00 50 03 01 00 00 00 01 00 01 00 00 00 00 ..X.....

Figure 98. Get_Attributes_All Response over TCP (Class 0x01, Instance 0x00)

B. CIP GET_ATTRIBUTE_LIST TEST CASES

This section shows the results of the CIP Get_Attributes_List test cases.

(1) T32 Results

A Get_Attribute_List command with a fuzzed Class field returns two different responses. Figure 99 shows a General Status 0x08 “Service not supported” [15] response and Figure 100 illustrates a General Status 0x05 “Path destination unknown” [15] response.

No.	Time	Source	Destination	Protocol	Length	Resp	Info
6	10:45:11.579748	192.168.0.62	192.168.0.59	ENIP	82		Register Session (Req), Session: 0x00000000
7	10:45:11.590003	192.168.0.59	192.168.0.62	ENIP	82		Register Session (Rsp), Session: 0x087F5FAE
8	10:45:11.592147	192.168.0.62	192.168.0.59	TCP	60		40450 → 44818 [ACK] Seq=29 Ack=29 Win=29280 Len=0
9	10:45:11.602894	192.168.0.62	192.168.0.59	CIP CM	140		Connection Manager – Forward Open (Message Router)
10	10:45:11.609789	192.168.0.59	192.168.0.62	CIP CM	124		Success: Connection Manager – Forward Open
11	10:45:11.620187	192.168.0.62	192.168.0.59	CIP	110		Identity – Get Attribute List
12	10:45:11.629986	192.168.0.59	192.168.0.62	CIP	104		Service not supported: Identity – Get Attribute List
▶ Ethernet II, Src: Rockwell_L1:28:4c (00:1d:9c:a1:28:4c), Dst: Vmware_56:20:17 (00:0c:29:56:20:17)							
▶ Internet Protocol Version 4, Src: 192.168.0.59, Dst: 192.168.0.62							
▶ Transmission Control Protocol, Src Port: 44818, Dst Port: 40450, Seq: 99, Ack: 171, Len: 50							
▶ EtherNet/IP (Industrial Protocol), Session: 0x087F5FAE, Send Unit Data							
▼ Common Industrial Protocol							
▼ Service: Get Attribute List (Response)							
1... = Request/Response: Response (0x1)							
.000 0011 = Service: Get Attribute List (0x03)							
▼ Status: Service not supported:							
General Status: Service not supported (0x08)							
Additional Status Size: 0 (words)							
[Request Path Size: 2 (words)]							
▼ [Request Path: Identity, Instance: 0x01]							
▶ [Path Segment: 0x20 (8-Bit Class Segment)]							
▶ [Path Segment: 0x24 (8-Bit Instance Segment)]							
0000	00 0c 29 56 20 17 00 1d	9c a1 28 4c 00 00 45 00	..)V(L..E.				
0010	00 5a 0d a9 00 00 00 06	a5 2b c0 a0 00 3b c0 a0	.Z.....+.....				
0020	00 3e af 12 9e 0a 4d 5f	73 63 a1 0a ac f2 5b 18	..,...M.sc...P.				
0030	07 d0 29 b0 00 00 70 00	1a 00 ae 5f 7f 00 00 00	...)..D.				
0040	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00				
0050	00 00 00 00 02 00 a1 00	04 00 01 00 fe 00 b1 00				
0060	06 00 00 00 03 00 00 00				

Figure 99. Get_Attribute_List Response over TCP (Class 0x01, Instance 0x01, Attribute 0x01)

No.	Time	Source	Destination	Protocol	Length	Resp	Info
5543	20:13:10.918742	192.168.0.59	192.168.0.62	CIP	104		Path destination unknown: CompoNet Repeater – Get Attribute List
5544	20:13:10.956148	192.168.0.62	192.168.0.59	TCP	60		58898 → 44818 [ACK] Seq=103157 Ack=92099 Win=28944 Len=0
5545	20:13:11.015325	192.168.0.62	192.168.0.59	CIP	110		Class (0x7f) – Get Attribute List
5546	20:13:11.030141	192.168.0.59	192.168.0.62	CIP	104		Path destination unknown: Class (0x7f) – Get Attribute List
5547	20:13:11.068410	192.168.0.62	192.168.0.59	TCP	60		58898 → 44818 [ACK] Seq=103213 Ack=92149 Win=28944 Len=0
▶ EtherNet/IP (Industrial Protocol), Session: 0x38D05C70, Send Unit Data							
▼ Common Industrial Protocol							
▼ Service: Get Attribute List (Response)							
1... = Request/Response: Response (0x1)							
.000 0011 = Service: Get Attribute List (0x03)							
▼ Status: Path destination unknown:							
General Status: Path destination unknown (0x05)							
Additional Status Size: 0 (words)							
[Request Path Size: 2 (words)]							
▼ [Request Path: Class: 0x7F, Instance: 0x01]							
▼ [Path Segment: 0x20 (8-Bit Class Segment)]							
[001. = Path Segment Type: Logical Segment (1)]							
[...0 00.. = Logical Segment Type: Class ID (0)]							
[.... ..00 = Logical Segment Format: 8-bit Logical Segment (0)]							
▼ [8-Bit Class Segment]							
[Class: Unknown (0x7f)]							
▼ [Path Segment: 0x24 (8-Bit Instance Segment)]							
[001. = Path Segment Type: Logical Segment (1)]							
[...0 01.. = Logical Segment Type: Instance ID (1)]							
[.... ..00 = Logical Segment Format: 8-bit Logical Segment (0)]							
▼ [8-Bit Instance Segment]							
[Instance: 0x01]							
0000	00 0c 29 56 20 17 00 1d	9c a1 28 4c 00 00 45 00	..)V(L..E.				
0010	00 5a 07 4b 00 00 00 06	b1 09 c0 a0 00 3b c0 a0	.Z.K.....				
0020	00 3e af 12 e6 12 4d 54	74 d0 63 12 9b 0a 50 18	..,...MT.t.c...P.				
0030	07 d0 e2 f1 00 00 70 00	1a 00 70 5c dd 38 00 00	...)..P..P\..8..				
0040	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00				
0050	00 00 00 00 02 00 a1 00	04 00 01 00 fe 00 b1 00				
0060	06 00 30 07 83 00 05 000.....				

Figure 100. Get_Attribute_List Response over TCP (Class 0x7F, Instance 0x01, Attribute 0x01)

(2) T33 Results

The Get_Attribute_List with fuzzed Instance field requests return two different responses: General Status 0x08 “Service not supported” [15] responses (Figure 101) and General Status 0x05 “Path destination unknown” [15] responses (Figure 102).

No.	Time	Source	Destination	Protocol	Length	Resp	Info
279	20:19:07.331183	192.168.0.62	192.168.0.59	TCP	60		58980 → 44818 [ACK] Seq=5101 Ack=4549 Win=29200 Len=0
280	20:19:07.393551	192.168.0.62	192.168.0.59	CIP	110		Identity - Get Attribute List
281	20:19:07.401548	192.168.0.59	192.168.0.62	CIP	184		Service not supported: Identity - Get Attribute List
282	20:19:07.439164	192.168.0.62	192.168.0.59	TCP	60		58980 → 44818 [ACK] Seq=5157 Ack=4599 Win=29200 Len=0
283	20:19:07.498921	192.168.0.62	192.168.0.59	CIP	110		Identity - Get Attribute List

▶	EtherNet/IP (Industrial Protocol), Session: 0x4ASC9AFA, Send Unit Data
▼	Common Industrial Protocol
▼	Service: Get Attribute List (Response)
1... = Request/Response: Response (0x1)
0000	0011 = Service: Get Attribute List (0x03)
▼	Status: Service not supported:
	General Status: Service not supported (0x08)
	Additional Status Size: 0 (words)
	[Request Path Size: 2 (words)]
	[Request Path: Identity, Instance: 0x01]
▼	[Path Segment: 0x20 (8-Bit Class Segment)]
	[001. = Path Segment Type: Logical Segment (1)]
	[...0 00.. = Logical Segment Type: Class ID (0)]
	[.... ..00 = Logical Segment Format: 8-bit Logical Segment (0)]
▼	[8-Bit Class Segment]
	[Class: Identity (0x01)]
▼	[Path Segment: 0x24 (8-Bit Instance Segment)]
	[001. = Path Segment Type: Logical Segment (1)]
	[...0 01.. = Logical Segment Type: Instance ID (1)]
	[.... ..00 = Logical Segment Format: 8-bit Logical Segment (0)]
▼	[8-Bit Instance Segment]
	[Instance: 0x01]

0000	00 0c 29 56 20 17 00 1d 9c a1 28 4c 08 00 45 00	..)V(L..E.
0010	00 5a 10 c7 00 00 00 06 a8 0d c0 a8 00 3b c0 a8	.Z.....
0020	00 3e af 12 e5 14 49 0c 95 e2 ed 4e 29 43 50 18	..>....I.N)CP.
0030	07 d0 78 ef 00 00 70 00 1a 00 fa 9a 5c 4a 00 00	..X...B.\J..
0040	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0050	00 00 00 00 02 00 a1 00 04 00 01 00 fe 80 b1 00
0060	05 00 59 00 83 00 00 00	..Y.....

Figure 101. Get_Attribute_List “Service Not Supported” Response over TCP (Class 0x01, Instance 0x01, Attribute 0x01)

No.	Time	Source	Destination	Protocol	Length	Resp	Info
5072	20:22:05.247740	192.168.0.59	192.168.0.62	CIP	104		Path destination unknown: Identity - Get Attribute List
5073	20:22:05.290613	192.168.0.62	192.168.0.59	TCP	60	58900 → 44818 [ACK] Seq=94421 Ack=84299 Win=28944 Len=0	
5074	20:22:05.345307	192.168.0.62	192.168.0.59	CIP	110		Identity - Get Attribute List
5075	20:22:05.357984	192.168.0.59	192.168.0.62	CIP	104		Path destination unknown: Identity - Get Attribute List
5076	20:22:05.394653	192.168.0.62	192.168.0.59	TCP	60	58900 → 44818 [ACK] Seq=94477 Ack=84349 Win=28944 Len=0	

▶ Frame 5075: 104 bytes on wire (832 bits), 104 bytes captured (832 bits) on interface 0
 ▶ Ethernet II, Src: Rockwell_al:28:4c (00:1d:9c:a1:28:4c), Dst: Vmware_56:20:17 (00:0c:29:56:20:17)
 ▶ Internet Protocol Version 4, Src: 192.168.0.59, Dst: 192.168.0.62
 ▶ Transmission Control Protocol, Src Port: 44818, Dst Port: 58900, Seq: 84299, Ack: 94477, Len: 50
 ▶ EtherNet/IP (Industrial Protocol), Session: 0x4A5C9AFA, Send Unit Data
 ▼ Common Industrial Protocol
 ▼ Service: Get Attribute List (Response)
 1... = Request/Response: Response (0x1)
 .000 0011 = Service: Get Attribute List (0x03)
 ▼ Status: Path destination unknown:
 General Status: Path destination unknown (0x05)
 Additional Status Size: 0 (words)
 [Request Path Size: 2 (words)]
 ▼ [Request Path: Identity, Instance: 0x01]
 ▼ [Path Segment: 0x20 (8-Bit Class Segment)]
 [001. = Path Segment Type: Logical Segment (1)]
 [... 0 00.. = Logical Segment Type: Class ID (0)]
 [.... ..00 = Logical Segment Format: 8-bit Logical Segment (0)]
 ▼ [8-Bit Class Segment]
 [Class: Identity (0x01)]
 ▼ [Path Segment: 0x24 (8-Bit Instance Segment)]
 [001. = Path Segment Type: Logical Segment (1)]
 [... 0 01.. = Logical Segment Type: Instance ID (1)]
 [.... ..00 = Logical Segment Format: 8-bit Logical Segment (0)]
 ▼ [8-Bit Instance Segment]
 [Instance: 0x01]
 0000 00 0c 29 56 20 17 00 1d 9c a1 28 4c 00 00 45 00 ..)V(L..E.
 0010 00 5a 17 02 00 00 00 06 a1 d2 c0 a0 00 3b c0 a0 .Z.....;..
 0020 00 3e af 12 e6 14 49 0d cd 68 ed 4f 06 2b 50 18 .>....I. .h.O.+P.
 0030 07 d0 ac 78 00 00 70 00 1a 00 fa 9a 5c 4a 00 00 ...x..p.J..
 0040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 0050 00 00 00 00 02 00 a1 00 04 00 01 00 fe 80 b1 00
 0060 06 00 94 06 83 00 05 00

Figure 102. Get_Attribute_List “Path Destination Unknown” Response over TCP (Class 0x01, Instance 0x01, Attribute 0x01)

(3) T34 Results

The Get_Attribute_List command with a fuzzed Attribute field returns General Status 0x08 “Service not supported” [15] responses as shown in Figure 103.

No.	Time	Source	Destination	Protocol	Length	Resp	Info
22	10:35:34.051794	192.168.0.59	192.168.0.62	CIP	104		Service not supported: Identity - Get Attribute List
23	10:35:34.054111	192.168.0.62	192.168.0.59	CIP	110		Identity - Get Attribute List
24	10:35:34.061960	192.168.0.59	192.168.0.62	CIP	104		Service not supported: Identity - Get Attribute List
25	10:35:34.062331	192.168.0.62	192.168.0.59	CIP	110		Identity - Get Attribute List
26	10:35:34.071368	192.168.0.59	192.168.0.62	CIP	104		Service not supported: Identity - Get Attribute List

▶	Frame 24: 104 bytes on wire (832 bits), 104 bytes captured (832 bits) on interface 0
▶	Ethernet II, Src: Rockwell_al:28:4c (00:1d:9c:a1:28:4c), Dst: Vmware_56:20:17 (00:0c:29:56:20:17)
▶	Internet Protocol Version 4, Src: 192.168.0.59, Dst: 192.168.0.62
▶	Transmission Control Protocol, Src Port: 44818, Dst Port: 40452, Seq: 449, Ack: 563, Len: 50
▶	EtherNet/IP (Industrial Protocol), Session: 0x1C23E177, Send Unit Data
▼	Common Industrial Protocol
▼	Service: Get Attribute List (Response)
1... = Request/Response: Response (0x1)
.000	0011 = Service: Get Attribute List (0x03)
▼	Status: Service not supported:
General Status:	Service not supported (0x08)
Additional Status Size:	0 (words)
[Request Path Size: 2 (words)]	
▼	[Request Path: Identity, Instance: 0x01]
▼	[Path Segment: 0x20 (8-Bit Class Segment)]
[001. = Path Segment Type: Logical Segment (1)]
[...0	00.. = Logical Segment Type: Class ID (0)]
[....	..00 = Logical Segment Format: 8-bit Logical Segment (0)]
▼	[8-Bit Class Segment]
[Class:	Identity (0x01)]
▼	[Path Segment: 0x24 (8-Bit Instance Segment)]
[001. = Path Segment Type: Logical Segment (1)]
[...0	01.. = Logical Segment Type: Instance ID (1)]
[....	..00 = Logical Segment Format: 8-bit Logical Segment (0)]
▼	[8-Bit Instance Segment]
[Instance:	0x01]

0000	00 0c 29 56 20 17 00 1d	9c a1 28 4c 00 00 45 00	..)V(L..E.
0010	00 5a 09 db 00 00 00 06	ae f9 c0 a0 00 3b c0 a0	.Z.....:..
0020	00 3e af 12 9e 04 49 16	6c 3d 6a 8d 54 e8 00 18	.>....I. lmj.T.P.
0030	07 d0 4f 9f 00 00 70 00	1a 00 77 e1 23 1c 00 00	..0...p. ..w.#...
0040	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
0050	00 00 00 00 02 00 a1 00	04 00 01 00 fe 80 b1 00
0060	06 00 07 00 03 00 00 00	

Figure 103. Get_Attribute_List “Service Not Supported” Response over TCP (Class 0x01, Instance 0x01, Attribute 200)

(4) T35 Results

A Get_Attribute_List request with the Attribute_count set to 223 is illustrated in Figure 104. Figure 105 shows the SUT response. Get_Attribute_List commands with the Attribute_count field exceeding 223 (Figure 106) receive a TCP ACK response (Figure 107).

No.	Time	Source	Destination	Protocol	Length	Resp	Info
509	21:35:33.010957	192.168.0.62	192.168.0.59	CIP	554		Identity - Get Attribute List
510	21:35:33.013829	192.168.0.59	192.168.0.62	CIP	184		Service not supported: Identity - Get Attribute List
511	21:35:33.050956	192.168.0.62	192.168.0.59	TCP	60		58960 → 44818 [ACK] Seq=617 Ack=149 Win=29200 Len=0
512	21:35:33.116313	192.168.0.62	192.168.0.59	TCP	60		58960 → 44818 [RST, ACK] Seq=617 Ack=149 Win=29200 Len=0
Common Industrial Protocol							
Service: Get Attribute List (Request)							
Request Path Size: 2 (words)							
Request Path: Identity, Instance: 0x01							
Get Attribute List (Request)							
Attribute Count: 223							
Attribute List							
Attribute: 1 (Vendor ID)							
Attribute: 2 (Device Type)							
Attribute: 3 (Product Code)							
Attribute: 4 (Revision)							
Attribute: 5 (Status)							
Attribute: 6 (Serial Number)							
Attribute: 7 (Product Name)							
Attribute: 1 (Vendor ID)							
Attribute: 2 (Device Type)							
Attribute: 3 (Product Code)							
0000	00 1d 9c a1 28 4c 00 0c	29 56 20 17 08 00 45 00(L..)V ...E.				
0010	02 1c e6 13 40 00 00 06	d0 fe c0 a8 00 3e c0 a8	...@.@.>..				
0020	00 3b e6 50 af 12 07 0b	b9 6e 4d 63 7a 6a 50 18	..i.P.... nMczjP.				
0030	72 10 48 24 00 00 70 00	dc 01 14 5a 68 5a 00 00	r.HS...p. ...ZhZ..				
0040	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00				
0050	00 00 00 00 02 00 a1 00	04 00 72 58 5a 14 b1 00rXZ...				
0060	c8 01 00 00 03 02 20 01	24 01 df 00 01 00 02 00 \$......				
0070	03 00 04 00 05 00 06 00	07 00 01 00 02 00 03 00				
0080	04 00 05 00 06 00 07 00	01 00 02 00 03 00 04 00				
0090	05 00 06 00 07 00 01 00	02 00 03 00 04 00 05 00				
00a0	06 00 07 00 01 00 02 00	03 00 04 00 05 00 06 00				
00b0	07 00 01 00 02 00 03 00	04 00 05 00 06 00 07 00				
00c0	01 00 02 00 03 00 04 00	05 00 06 00 07 00 01 00				
00d0	02 00 03 00 04 00 05 00	06 00 07 00 01 00 02 00				
00e0	03 00 04 00 05 00 06 00	07 00 01 00 02 00 03 00				
00f0	04 00 05 00 06 00 07 00	01 00 02 00 03 00 04 00				
0100	05 00 06 00 07 00 01 00	02 00 03 00 04 00 05 00				
0110	06 00 07 00 01 00 02 00	03 00 04 00 05 00 06 00				
0120	07 00 01 00 02 00 03 00	04 00 05 00 06 00 07 00				
0130	01 00 02 00 03 00 04 00	05 00 06 00 07 00 01 00				
0140	02 00 03 00 04 00 05 00	06 00 07 00 01 00 02 00				
0150	03 00 04 00 05 00 06 00	07 00 01 00 02 00 03 00				
0160	04 00 05 00 06 00 07 00	01 00 02 00 03 00 04 00				
0170	05 00 06 00 07 00 01 00	02 00 03 00 04 00 05 00				
0180	06 00 07 00 01 00 02 00	03 00 04 00 05 00 06 00				
0190	07 00 01 00 02 00 03 00	04 00 05 00 06 00 07 00				
01a0	01 00 02 00 03 00 04 00	05 00 06 00 07 00 01 00				
01b0	02 00 03 00 04 00 05 00	06 00 07 00 01 00 02 00				
01c0	03 00 04 00 05 00 06 00	07 00 01 00 02 00 03 00				
01d0	04 00 05 00 06 00 07 00	01 00 02 00 03 00 04 00				
01e0	05 00 06 00 07 00 01 00	02 00 03 00 04 00 05 00				
01f0	06 00 07 00 01 00 02 00	03 00 04 00 05 00 06 00				
0200	07 00 01 00 02 00 03 00	04 00 05 00 06 00 07 00				
0210	01 00 02 00 03 00 04 00	05 00 06 00 07 00 01 00				
0220	02 00 03 00 04 00 05 00	06 00 07 00 01 00 02 00				

Figure 104. Get_Attribute_List Request over TCP (Attribute_count: 223)

No.	Time	Source	Destination	Protocol	Length	Resp	Info
509	21:35:33.010957	192.168.0.62	192.168.0.59	CIP	554		Identity - Get Attribute List
510	21:35:33.013829	192.168.0.59	192.168.0.62	CIP	184		Service not supported: Identity - Get Attribute List
511	21:35:33.050956	192.168.0.62	192.168.0.59	TCP	60		58960 → 44818 [ACK] Seq=617 Ack=149 Win=29200 Len=0
512	21:35:33.116313	192.168.0.62	192.168.0.59	TCP	60		58960 → 44818 [RST, ACK] Seq=617 Ack=149 Win=29200 Len=0
Common Industrial Protocol							
Service: Get Attribute List (Response)							
1... .. = Request/Response: Response (0x1)							
..000 0011 = Service: Get Attribute List (0x03)							
Status: Service not supported:							
General Status: Service not supported (0x00)							
Additional Status Size: 0 (words)							
[Request Path Size: 2 (words)]							
[Request Path: Identity, Instance: 0x01]							
[Path Segment: 0x20 (8-Bit Class Segment)]							
[001. = Path Segment Type: Logical Segment (1)]							
[...0 00.. = Logical Segment Type: Class ID (0)]							
[.... ..00 = Logical Segment Format: 8-bit Logical Segment (0)]							
[8-Bit Class Segment]							
[Class: Identity (0x01)]							
[Path Segment: 0x24 (8-Bit Instance Segment)]							
[001. = Path Segment Type: Logical Segment (1)]							
[...0 01.. = Logical Segment Type: Instance ID (1)]							
[.... ..00 = Logical Segment Format: 8-bit Logical Segment (0)]							
[8-Bit Instance Segment]							
[Instance: 0x01]							
0000	00 0c 29 56 20 17 00 1d	9c a1 28 4c 08 00 45 00	..)V(L..E.				
0010	00 5a 00 7b 00 00 00 06	b8 59 c0 a8 00 3b c0 a8	.Z.{.... .Y...j..				
0020	00 3e af 12 e6 59 4d 63	7a 6a 07 0b bb 62 58 18	.>...PMc zj...bP.				
0030	07 c0 17 2a 00 00 70 00	1a 00 14 5a 68 5a 00 00	...*.P. ...ZHZ..				
0040	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00				
0050	00 00 00 00 02 00 a1 00	04 00 01 00 fe 00 b1 00				
0060	06 00 00 00 03 00 08 00					

Figure 105. Get_Attribute_List Response over TCP (Attribute_count: 223)

No.	Time	Source	Destination	Protocol	Length	Resp	Info
525	21:36:02.882581	192.168.0.62	192.168.0.59	CIP CM	142		Connection Manager - Forward Open (Message Router)
526	21:36:02.893408	192.168.0.59	192.168.0.62	CIP CM	124		Success: Connection Manager - Forward Open
527	21:36:02.903549	192.168.0.62	192.168.0.59	CIP	556		Identity - Get Attribute List
528	21:36:03.043233	192.168.0.59	192.168.0.62	TCP	60		44818 → 58962 [ACK] Seq=99 Ack=619 Win=2000 Len=0
529	21:36:12.873195	Rockwell_a1:28:4c	Broadcast	ARP	60		Gratuitous ARP for 192.168.0.59 (Request)
530	21:36:13.073513	Rockwell_a1:28:4c	Broadcast	ARP	60		Gratuitous ARP for 192.168.0.59 (Request)
Ethernet/IP (Industrial Protocol), Session: 0x0CFF1D2A, Send Unit Data							
Common Industrial Protocol							
Service: Get Attribute List (Request)							
Request Path Size: 2 (words)							
Request Path: Identity, Instance: 0x01							
Get Attribute List (Request)							
Attribute Count: 224							
Attribute List							
Attribute: 1 (Vendor ID)							
Attribute: 2 (Device Type)							
Attribute: 3 (Product Code)							
Attribute: 4 (Revision)							
Attribute: 5 (Status)							
Attribute: 6 (Serial Number)							
Attribute: 7 (Product Name)							
0000	00 1d 9c a1 28 4c 08 0c	29 56 20 17 00 00 45 00(L..)V ...E.				
0010	02 1e 23 bd 40 00 00 06	93 53 c0 a8 00 3e c0 a8	..#.@.@. .S...>..				
0020	00 3b e6 52 af 12 1c cf	60 05 49 1c 4f ff 50 18	.;.R.... 'I.O.P.				
0030	72 10 74 d9 00 00 70 00	de 01 2a 1d ff 0c 00 00	r.t...P. ...*.....				
0040	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00				
0050	00 00 00 00 02 00 a1 00	04 00 3b 5b 1e 2a b1 00 ;j/k...*				
0060	c0 01 00 00 03 02 20 01	24 01 c0 00 01 00 02 00 S.....				
0070	03 00 04 00 05 00 06 00	07 00 01 00 02 00 03 00				
0080	04 00 05 00 06 00 07 00	01 00 02 00 03 00 04 00				
0090	05 00 06 00 07 00 01 00	02 00 03 00 04 00 05 00				
00a0	06 00 07 00 01 00 02 00	03 00 04 00 05 00 06 00				
00b0	07 00 01 00 02 00 03 00	04 00 05 00 06 00 07 00				
00c0	01 00 02 00 03 00 04 00	05 00 06 00 07 00 01 00				
00d0	02 00 03 00 04 00 05 00	06 00 07 00 01 00 02 00				
00e0	03 00 04 00 05 00 06 00	07 00 01 00 02 00 03 00				
00f0	04 00 05 00 06 00 07 00	01 00 02 00 03 00 04 00				
0100	05 00 06 00 07 00 01 00	02 00 03 00 04 00 05 00				
0110	06 00 07 00 01 00 02 00	03 00 04 00 05 00 06 00				
0120	07 00 01 00 02 00 03 00	04 00 05 00 06 00 07 00				
0130	01 00 02 00 03 00 04 00	05 00 06 00 07 00 01 00				
0140	02 00 03 00 04 00 05 00	06 00 07 00 01 00 02 00				
0150	03 00 04 00 05 00 06 00	07 00 01 00 02 00 03 00				
0160	04 00 05 00 06 00 07 00	01 00 02 00 03 00 04 00				
0170	05 00 06 00 07 00 01 00	02 00 03 00 04 00 05 00				
0180	06 00 07 00 01 00 02 00	03 00 04 00 05 00 06 00				
0190	07 00 01 00 02 00 03 00	04 00 05 00 06 00 07 00				
01a0	01 00 02 00 03 00 04 00	05 00 06 00 07 00 01 00				
01b0	02 00 03 00 04 00 05 00	06 00 07 00 01 00 02 00				
01c0	03 00 04 00 05 00 06 00	07 00 01 00 02 00 03 00				
01d0	04 00 05 00 06 00 07 00	01 00 02 00 03 00 04 00				
01e0	05 00 06 00 07 00 01 00	02 00 03 00 04 00 05 00				
01f0	06 00 07 00 01 00 02 00	03 00 04 00 05 00 06 00				
0200	07 00 01 00 02 00 03 00	04 00 05 00 06 00 07 00				
0210	01 00 02 00 03 00 04 00	05 00 06 00 07 00 01 00				
0220	02 00 03 00 04 00 05 00	06 00 07 00				

Figure 106. Get_Attribute_List Request over TCP (Attribute_count: 224)

No.	Time	Source	Destination	Protocol	Length	Resp	Info
525	21:36:02.882501	192.168.0.62	192.168.0.59	CIP CM	142		Connection Manager - Forward Open (Message Router)
526	21:36:02.893488	192.168.0.59	192.168.0.62	CIP CM	124		Success: Connection Manager - Forward Open
527	21:36:02.903549	192.168.0.62	192.168.0.59	CIP	556		Identity - Get Attribute List
528	21:36:03.043233	192.168.0.59	192.168.0.62	TCP	60	44818 → 58962 [ACK] Seq=99 Ack=619 Win=2000 Len=0	
529	21:36:12.873195	Rockwell_a1:28:4c	Broadcast	ARP	60		Gratuitous ARP for 192.168.0.59 (Request)
530	21:36:13.073513	Rockwell_a1:28:4c	Broadcast	ARP	60		Gratuitous ARP for 192.168.0.59 (Request)


```

▶ Frame 528: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
▶ Ethernet II, Src: Rockwell_a1:28:4c (00:1d:9c:a1:28:4c), Dst: Vmware_56:20:17 (00:0c:29:56:20:17)
▶ Internet Protocol Version 4, Src: 192.168.0.59, Dst: 192.168.0.62
▼ Transmission Control Protocol, Src Port: 44818, Dst Port: 58962, Seq: 99, Ack: 619, Len: 0
  Source Port: 44818
  Destination Port: 58962
  [Stream index: 28]
  [TCP Segment Len: 0]
  Sequence number: 99 (relative sequence number)
  Acknowledgment number: 619 (relative ack number)
  Header Length: 20 bytes
  ▶ Flags: 0x010 (ACK)
  Window size value: 2000
  [Calculated window size: 2000]
  [Window size scaling factor: -2 (no window scaling used)]
  Checksum: 0x78ef [unverified]
  [Checksum Status: Unverified]
  Urgent pointer: 0
  ▶ [SEQ/ACK analysis]

```



```

0000 00 0c 29 56 20 17 00 1d 9c a1 28 4c 00 00 45 00 ..)V ... ..(L..E.
0010 00 28 00 7f 00 00 00 06 b8 07 c0 a8 00 3b c0 a8 .(..... ..;..
0020 00 3e af 12 e6 52 49 1c 4f ff 1c cf 61 fb 50 18 .>...RI. 0...a.P.
0030 07 d0 78 ef 00 00 00 00 00 00 00 00 ..x.....

```

Figure 107. Get_Attribute_List Response over TCP (Attribute_count: 224)

C. CIP GET_ATTRIBUTE_SINGLE TEST CASES

This section shows the results of the CIP Get_Attributes_Single test cases.

(1) T36 Results

Get_Attribute_Single with a fuzzed Class field returns either a “Service not supported” response (Figure 108) or a “Path destination unknown response” (Figure 109).

No.	Time	Source	Destination	Protocol	Length	Resp	Info
747	21:58:36.773070	192.168.0.62	192.168.0.59	CIP	108		Identity - Get Attribute Single
748	21:58:36.780719	192.168.0.59	192.168.0.62	CIP	104		Service not supported: Identity - Get Attribute Single
749	21:58:36.786856	192.168.0.62	192.168.0.59	CIP	108		Identity - Get Attribute Single
750	21:58:36.799687	192.168.0.59	192.168.0.62	CIP	104		Path destination unknown: Identity - Get Attribute Single


```

▶ Frame 748: 104 bytes on wire (832 bits), 104 bytes captured (832 bits) on interface 0
▶ Ethernet II, Src: Rockwell_a1:28:4c (00:1d:9c:a1:28:4c), Dst: Vmware_56:20:17 (00:0c:29:56:20:17)
▶ Internet Protocol Version 4, Src: 192.168.0.59, Dst: 192.168.0.62
▶ Transmission Control Protocol, Src Port: 44818, Dst Port: 42048, Seq: 18499, Ack: 28041, Len: 50
▶ Ethernet/IP (Industrial Protocol), Session: 0x16B06CC0, Send Unit Data
▼ Common Industrial Protocol
  ▶ Service: Get Attribute Single (Response)
  1..... = Request/Response: Response (0x1)
  .000 1110 = Service: Get Attribute Single (0x0e)
  ▶ Status: Service not supported:
    General Status: Service not supported (0x00)
    Additional Status Size: 0 (words)
    [Request Path Size: 2 (words)]
  ▶ [Request Path: Identity, Instance: 0x00]
    ▶ [Path Segment: 0x20 (8-Bit Class Segment)]
      [001. .... = Path Segment Type: Logical Segment (1)]
      [...0 00.. = Logical Segment Type: Class ID (0)]
      [.... ..00 = Logical Segment Format: 8-bit Logical Segment (0)]
      ▶ [8-Bit Class Segment]
        [Class: Identity (0x01)]
      ▶ [Path Segment: 0x24 (8-Bit Instance Segment)]
        [001. .... = Path Segment Type: Logical Segment (1)]
        [...0 01.. = Logical Segment Type: Instance ID (1)]
        [.... ..00 = Logical Segment Format: 8-bit Logical Segment (0)]
        ▶ [8-Bit Instance Segment]
          [Instance: 0x00]

```



```

0000 00 0c 29 56 20 17 00 1d 9c a1 28 4c 00 00 45 00 ..)V ... ..(L..E.
0010 00 5a 92 00 00 00 00 06 26 54 c0 a8 00 3b c0 a8 .Z..... &T...;..
0020 00 3e af 12 a4 40 49 34 68 bf b3 0b 76 00 50 18 .>...@I4 h...v.P.
0030 07 d0 90 a5 00 00 70 00 1a 00 c0 6c b0 16 00 00 .....p. ...l....
0040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0050 00 00 00 00 02 00 a1 00 04 00 01 00 fe 00 b1 00 .....
0060 06 00 71 01 8e 00 00 00 ..q.....

```

Figure 108. Get_Attribute_Single “Service Not Supported” Response over TCP

No.	Time	Source	Destination	Protocol	Length	Resp	Info
9	21:54:23.413966	192.168.0.62	192.168.0.59	CIP	108		Class (0x76) - Get Attribute Single
10	21:54:23.424769	192.168.0.59	192.168.0.62	CIP	104		Path destination unknown: Class (0x76) - Get Attribute Single
11	21:54:23.429957	192.168.0.62	192.168.0.59	CIP	108		Class (0x86) - Get Attribute Single
12	21:54:23.434189	192.168.0.59	192.168.0.62	CIP	104		Path destination unknown: Class (0x86) - Get Attribute Single

▼ Common Industrial Protocol
▼ Service: Get Attribute Single (Response)
1... = Request/Response: Response (0x1)
.000 1110 = Service: Get Attribute Single (0x0e)
▼ Status: Path destination unknown:
General Status: Path destination unknown (0x05)
Additional Status Size: 0 (words)
[Request Path Size: 2 (words)]
[Request Path: Class: 0x76, Instance: 0x00]
▼ [Path Segment: 0x20 (8-Bit Class Segment)]
[001. = Path Segment Type: Logical Segment (1)]
[...0 00.. = Logical Segment Type: Class ID (0)]
[.... ..00 = Logical Segment Format: 8-bit Logical Segment (0)]
▼ [8-Bit Class Segment]
[Class: Unknown (0x76)]
▼ [Path Segment: 0x24 (8-Bit Instance Segment)]
[001. = Path Segment Type: Logical Segment (1)]
[...0 01.. = Logical Segment Type: Instance ID (1)]
[.... ..00 = Logical Segment Format: 8-bit Logical Segment (0)]
▼ [8-Bit Instance Segment]
[Instance: 0x00]

0000	00	0c	29	56	20	17	00	1d	9c	a1	28	4c	08	00	45	00	..)V(L..E.
0010	00	5a	89	92	00	00	80	86	2f	42	c0	a0	00	3b	c0	a8	.Z...../B....
0020	00	3e	af	12	a4	3c	49	31	a3	ce	8f	cc	70	17	50	18	.>...<11p.P.
0030	07	d0	3c	1e	00	00	70	00	1a	00	9d	2d	90	1e	00	00	..<...p.
0040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0050	00	00	00	00	02	00	a1	00	04	00	01	00	fe	00	b1	00
0060	06	00	01	00	8e	00	05	00								

Figure 109. Get_Attribute_Single “Path Destination Unknown” Response over TCP

(2) T37 Results

The Get_Attribute_Single command returns an “Attribute not supported” response when the Instance field is set to 0x00 and Class and Attribute fields are 0x01 (Figure 110). When the Instance field is 0x01, with the same Class and Attribute fields, the SUT returns a “Service not supported” message (Figure 111). All other Instance fields with the Class and Attribute fields set to 0x01 return “Path destination unknown” (Figure 112).

No.	Time	Source	Destination	Protocol	Length	Resp	Info
1939	22:06:29.151159	192.168.0.62	192.168.0.59	CIP	108		Identity - Get Attribute Single
1940	22:06:29.168416	192.168.0.59	192.168.0.62	CIP	104		Attribute not supported: Identity - Get Attribute Single
1941	22:06:29.171582	192.168.0.62	192.168.0.59	CIP	108		Identity - Get Attribute Single
1942	22:06:29.188215	192.168.0.59	192.168.0.62	CIP	104		Attribute not supported: Identity - Get Attribute Single

Transmission Control Protocol, Src Port: 44818, Dst Port: 42054, Seq: 48299, Ack: 52225, Len: 50
EtherNet/IP (Industrial Protocol), Session: 0x08643D7D, Send Unit Data
Common Industrial Protocol
Service: Get Attribute Single (Response)
1... = Request/Response: Response (0x1)
.000 1110 = Service: Get Attribute Single (0x0e)
Status: Attribute not supported:
General Status: Attribute not supported (0x14)
Additional Status Size: 0 (words)
[Request Path Size: 2 (words)]
[Request Path: Identity, Instance: 0x01]
[Path Segment: 0x20 (8-Bit Class Segment)]
[001. = Path Segment Type: Logical Segment (1)]
[...0 00.. = Logical Segment Type: Class ID (0)]
[.... ..00 = Logical Segment Format: 8-bit Logical Segment (0)]
[8-Bit Class Segment]
[Class: Identity (0x01)]
[Path Segment: 0x24 (8-Bit Instance Segment)]
[001. = Path Segment Type: Logical Segment (1)]
[...0 01.. = Logical Segment Type: Instance ID (1)]
[.... ..00 = Logical Segment Format: 8-bit Logical Segment (0)]
[8-Bit Instance Segment]
[Instance: 0x01]

```

0000 00 0c 29 56 20 17 00 1d 9c a1 28 4c 08 00 45 00 ..)V ... ..(L..E.
0010 00 5a 97 03 00 00 00 06 21 d1 c0 a8 00 3b c0 a8 .Z.....!....;..
0020 00 3e af 12 a4 46 4d 82 f1 f8 6f dc 62 97 50 18 .>...FM. ..o.b.P.
0030 07 d0 90 e8 00 00 70 00 1a 00 7d 3d 64 0b 00 00 .....p. ..}=d...
0040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0050 00 00 00 02 00 a1 00 04 00 01 00 fe 80 b1 00 .....
0060 06 00 c5 03 8e 00 14 00 .....

```

Figure 110. Get_Attribute_Single “Attribute Not Supported” Response over TCP

No.	Time	Source	Destination	Protocol	Length	Resp	Info
745	21:58:36.749219	192.168.0.62	192.168.0.59	CIP	108		Identity - Get Attribute Single
746	21:58:36.759512	192.168.0.59	192.168.0.62	CIP	104		Path destination unknown: Identity - Get Attribute Single
747	21:58:36.773070	192.168.0.62	192.168.0.59	CIP	108		Identity - Get Attribute Single
748	21:58:36.780719	192.168.0.59	192.168.0.62	CIP	104		Service not supported: Identity - Get Attribute Single
749	21:58:36.786856	192.168.0.62	192.168.0.59	CIP	108		Identity - Get Attribute Single
750	21:58:36.799687	192.168.0.59	192.168.0.62	CIP	104		Path destination unknown: Identity - Get Attribute Single
751	21:58:36.808064	192.168.0.62	192.168.0.59	CIP	108		Identity - Get Attribute Single

Common Industrial Protocol
Service: Get Attribute Single (Response)
1... = Request/Response: Response (0x1)
.000 1110 = Service: Get Attribute Single (0x0e)
Status: Service not supported:
General Status: Service not supported (0x00)
Additional Status Size: 0 (words)
[Request Path Size: 2 (words)]
[Request Path: Identity, Instance: 0x00]
[Path Segment: 0x20 (8-Bit Class Segment)]
[001. = Path Segment Type: Logical Segment (1)]
[...0 00.. = Logical Segment Type: Class ID (0)]
[.... ..00 = Logical Segment Format: 8-bit Logical Segment (0)]
[8-Bit Class Segment]
[Class: Identity (0x01)]
[Path Segment: 0x24 (8-Bit Instance Segment)]
[001. = Path Segment Type: Logical Segment (1)]
[...0 01.. = Logical Segment Type: Instance ID (1)]
[.... ..00 = Logical Segment Format: 8-bit Logical Segment (0)]
[8-Bit Instance Segment]
[Instance: 0x00]

```

0000 00 0c 29 56 20 17 00 1d 9c a1 28 4c 08 00 45 00 ..)V ... ..(L..E.
0010 00 5a 92 00 00 00 00 06 26 54 c0 a8 00 3b c0 a8 .Z.....67....;..
0020 00 3e af 12 a4 40 49 34 68 bf b3 0b 76 00 50 18 .>...@14 h...v.P.
0030 07 d0 98 a5 00 00 70 00 1a 00 c0 6c b0 16 00 00 .....p. ...l....
0040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0050 00 00 00 02 00 a1 00 04 00 01 00 fe 80 b1 00 .....
0060 06 00 71 01 8e 00 08 00 .....
..q.....

```

Figure 111. Get_Attribute_Single “Service Not Supported” Response over TCP

No.	Time	Source	Destination	Protocol	Length	Resp	Info
	389	21:58:33.588220	192.168.0.62	192.168.0.59	CIP	108	Identity - Get Attribute Single
	390	21:58:33.599551	192.168.0.59	192.168.0.62	CIP	104	Path destination unknown: Identity - Get Attribute Single
	391	21:58:33.612740	192.168.0.62	192.168.0.59	CIP	108	Identity - Get Attribute Single
	392	21:58:33.619795	192.168.0.59	192.168.0.62	CIP	104	Path destination unknown: Identity - Get Attribute Single
	393	21:58:33.632275	192.168.0.62	192.168.0.59	CIP	108	Identity - Get Attribute Single
	394	21:58:33.639988	192.168.0.59	192.168.0.62	CIP	104	Path destination unknown: Identity - Get Attribute Single
	395	21:58:33.647072	192.168.0.62	192.168.0.59	CIP	108	Identity - Get Attribute Single
▼ Common Industrial Protocol							
▼ Service: Get Attribute Single (Response)							
1... = Request/Response: Response (0x1)							
.000 1110 = Service: Get Attribute Single (0x0e)							
▼ Status: Path destination unknown:							
General Status: Path destination unknown (0x05)							
Additional Status Size: 0 (words)							
[Request Path Size: 2 (words)]							
▼ [Request Path: Identity, Instance: 0x25]							
▼ [Path Segment: 0x20 (8-Bit Class Segment)]							
[001. = Path Segment Type: Logical Segment (1)]							
[...0 00.. = Logical Segment Type: Class ID (0)]							
[.... ..00 = Logical Segment Format: 8-bit Logical Segment (0)]							
▼ [8-Bit Class Segment]							
[Class: Identity (0x01)]							
▼ [Path Segment: 0x24 (8-Bit Instance Segment)]							
[001. = Path Segment Type: Logical Segment (1)]							
[...0 01.. = Logical Segment Type: Instance ID (1)]							
[.... ..00 = Logical Segment Format: 8-bit Logical Segment (0)]							
▼ [8-Bit Instance Segment]							
[Instance: 0x25]							
0000	00 0c 29 56 20 17 00 1d	9c a1 28 4c 08 00 45 00	..)V(L..E.				
0010	00 5a 91 ce 00 00 00 06	27 06 c0 a0 00 3b c0 a8	.Z.....'.....				
0020	00 3e af 12 a4 40 49 34	45 fb b3 0b 50 74 50 18	.>...@I4 E...PtP.				
0030	07 d0 95 f6 00 00 70 00	1a 00 c0 6c b0 16 00 00P. ...l....				
0040	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00				
0050	00 00 00 00 02 00 a1 00	04 00 01 00 fe 80 b1 00				
0060	06 00 bf 00 0e 00 05 00					

Figure 112. Get_Attribute_Single “Path Destination Unknown” Response over TCP

(3) T38 Results

The MicroLogix responds to all Get_Attribute_Single requests with a fuzzed Attribute field and the Class and Instance fields set to 0x00 with an “Attribute not supported” message (Figure 113).

No.	Time	Source	Destination	Protocol	Length	Resp	Info
12	22:06:11.510804	192.168.0.59	192.168.0.62	CIP	104		Attribute not supported: Identity - Get Attribute Single
13	22:06:11.521345	192.168.0.62	192.168.0.59	CIP	108		Identity - Get Attribute Single
14	22:06:11.530498	192.168.0.59	192.168.0.62	CIP	104		Attribute not supported: Identity - Get Attribute Single
15	22:06:11.536202	192.168.0.62	192.168.0.59	CIP	108		Identity - Get Attribute Single
16	22:06:11.540333	192.168.0.59	192.168.0.62	CIP	104		Attribute not supported: Identity - Get Attribute Single
17	22:06:11.546259	192.168.0.62	192.168.0.59	CIP	108		Identity - Get Attribute Single
18	22:06:11.560740	192.168.0.59	192.168.0.62	CIP	104		Attribute not supported: Identity - Get Attribute Single
▼ Common Industrial Protocol							
▼ Service: Get Attribute Single (Response)							
1... = Request/Response: Response (0x1)							
.000 1110 = Service: Get Attribute Single (0x0e)							
▼ Status: Attribute not supported:							
General Status: Attribute not supported (0x14)							
Additional Status Size: 0 (words)							
[Request Path Size: 2 (words)]							
▼ [Request Path: Identity, Instance: 0x01]							
▼ [Path Segment: 0x20 (8-Bit Class Segment)]							
[001. = Path Segment Type: Logical Segment (1)]							
[...0 00.. = Logical Segment Type: Class ID (0)]							
[.... ..00 = Logical Segment Format: 8-bit Logical Segment (0)]							
▼ [8-Bit Class Segment]							
[Class: Identity (0x01)]							
▼ [Path Segment: 0x24 (8-Bit Instance Segment)]							
[001. = Path Segment Type: Logical Segment (1)]							
[...0 01.. = Logical Segment Type: Instance ID (1)]							
[.... ..00 = Logical Segment Format: 8-bit Logical Segment (0)]							
▼ [8-Bit Instance Segment]							
[Instance: 0x01]							
0000	00 0c 29 56 20 17 00 1d	9c a1 28 4c 08 00 45 00	..)V(L..E.				
0010	00 5a 93 41 00 00 00 06	25 93 c0 a8 00 3b c0 a8	.Z.A....%.....				
0020	00 3e af 12 a4 46 4d 82	36 14 6f db 97 ab 50 18	.>...FM. 6.O...P.				
0030	07 d0 d9 bd 00 00 70 00	1a 00 7d 3d 64 0b 00 00P. ..)md...				
0040	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00				
0050	00 00 00 00 02 00 a1 00	04 00 01 00 fe 80 b1 00				
0060	06 00 03 00 0e 00 14 00					

Figure 113. Get_Attribute_Single “Attribute Not Supported” Response over TCP

D. CIP_FIND_NEXT_OBJECT_INSTANCE TEST CASES

This section shows the results of the CIP Find_Next_Object_Instance test cases.

(1) T39 Results

The CIP Find_Next_Object_Instance command with a fuzzed Class field returns “Service not supported” for six Class field inputs, as illustrated by Figure 114. All other fuzzed Classes returned “Path destination unknown” responses (Figure 115).

No.	Time	Source	Destination	Protocol	Length	Resp	Info
17	21:53:39.765893	192.168.0.62	192.168.0.59	CIP	101		Identity - Find Next Object Instance
18	21:53:39.772972	192.168.0.59	192.168.0.62	CIP	100		Service not supported: Identity - Find Next Object Instance
19	21:53:39.813426	192.168.0.62	192.168.0.59	TCP	60		41884 → 44818 [ACK] Seq=76 Ack=75 Win=29200 Len=0
20	21:53:39.867620	192.168.0.62	192.168.0.59	TCP	60		41884 → 44818 [RST, ACK] Seq=76 Ack=75 Win=29200 Len=0

Common Industrial Protocol

Service: Find Next Object Instance (Response)

1... = Request/Response: Response (0x1)

.001 0001 = Service: Find Next Object Instance (0x11)

Status: Service not supported:

General Status: Service not supported (0x08)

Additional Status Size: 1 (words)

Additional Status

Additional Status: 0x0000

[Request Path Size: 2 (words)]

[Request Path: Identity, Instance: 0x00]

[Path Segment: 0x20 (8-Bit Class Segment)]

[001. = Path Segment Type: Logical Segment (1)]

[...0 00.. = Logical Segment Type: Class ID (0)]

[.... ..00 = Logical Segment Format: 8-bit Logical Segment (0)]

[8-Bit Class Segment]

[Class: Identity (0x01)]

[Path Segment: 0x24 (8-Bit Instance Segment)]

[001. = Path Segment Type: Logical Segment (1)]

[...0 01.. = Logical Segment Type: Instance ID (1)]

[.... ..00 = Logical Segment Format: 8-bit Logical Segment (0)]

[8-Bit Instance Segment]

[Instance: 0x00]

0000 00 0c 29 56 20 17 00 1d 9c a1 28 4c 08 00 45 00 ..)V(L..E.
0010 00 56 01 83 00 00 08 06 b7 55 c0 a8 00 3b c0 a8 .V.....U....
0020 00 3e af 12 a3 9c 4d 06 6b 25 17 da b4 c7 50 18 .>...M. k%...P.
0030 07 d0 66 f4 00 00 6f 00 16 00 25 2f e9 dd 00 00 ..f...o. .%/....
0040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0050 00 00 00 04 02 00 00 00 00 00 b2 00 06 00 91 00
0058 00 01 00 00

Figure 114. Find_Next_Object_Instance “Service Not Supported” Response over TCP

120

No.	Time	Source	Destination	Protocol	Length	Resp	Info
9	13:56:49.398348	192.168.0.62	192.168.0.59	CIP	100		Command Block - Find Next Object Instance
10	13:56:49.410004	192.168.0.59	192.168.0.62	CIP	100		Path destination unknown: Command Block - Find Next Object Instance
11	13:56:49.453155	192.168.0.62	192.168.0.59	TCP	60		50896 → 44818 [ACK] Seq=75 Ack=75 Win=29200 Len=0
12	13:56:49.504798	192.168.0.62	192.168.0.59	CIP	100		S-Analog Sensor - Find Next Object Instance
13	13:56:49.511078	192.168.0.59	192.168.0.62	CIP	100		Path destination unknown: S-Analog Sensor - Find Next Object Instance
14	13:56:49.512059	192.168.0.62	192.168.0.59	TCP	60		50896 → 44818 [ACK] Seq=121 Ack=121 Win=29200 Len=0

▶ Frame 10: 100 bytes on wire (800 bits), 100 bytes captured (800 bits) on interface 0
 ▶ Ethernet II, Src: Rockwell_l1:28:4c (00:1d:9c:a1:28:4c), Dst: Vmware_56:20:17 (00:0c:29:56:20:17)
 ▶ Internet Protocol Version 4, Src: 192.168.0.59, Dst: 192.168.0.62
 ▶ Transmission Control Protocol, Src Port: 44818, Dst Port: 50896, Seq: 29, Ack: 75, Len: 46
 ▶ EtherNet/IP (Industrial Protocol), Session: 0x0024718E, Send RR Data
 ▼ Common Industrial Protocol
 ▼ Service: Find Next Object Instance (Response)
 1... .. = Request/Response: Response (0x1)
 .001 0001 = Service: Find Next Object Instance (0x11)
 ▼ Status: Path destination unknown:
 General Status: Path destination unknown (0x05)
 Additional Status Size: 1 (words)
 ▼ Additional Status
 Additional Status: 0x0000
 [Request Path Size: 2 (words)]
 ▼ [Request Path: Command Block, Instance: 0x01]
 ▼ [Path Segment: 0x20 (8-Bit Class Segment)]
 [001. = Path Segment Type: Logical Segment (1)]
 [...0 00.. = Logical Segment Type: Class ID (0)]
 [.... ..00 = Logical Segment Format: 8-bit Logical Segment (0)]
 ▼ [8-Bit Class Segment]
 [Class: Command Block (0x27)]
 ▼ [Path Segment: 0x24 (8-Bit Instance Segment)]
 [001. = Path Segment Type: Logical Segment (1)]
 [...0 01.. = Logical Segment Type: Instance ID (1)]
 [.... ..00 = Logical Segment Format: 8-bit Logical Segment (0)]
 ▼ [8-Bit Instance Segment]
 [Instance: 0x01]

```

0000 00 0c 29 56 20 17 00 1d 9c a1 28 4c 00 00 45 00 ..)V ... ..(L..E.
0010 00 56 05 c0 00 00 00 06 b3 00 c0 a8 00 3b c0 a8 .V.....:..
0020 00 3e af 12 c6 00 49 18 00 ff 81 20 3e b0 50 18 .>...E: ...%P.
0030 07 d0 8e b3 00 00 6f 00 16 00 8e 71 24 0d 00 00 .....0. ...q$.
0040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0050 00 00 00 04 02 00 00 00 00 00 b2 00 06 00 91 00 .....
0060 05 01 00 00 .....

```

Figure 115. Find_Next_Object_Instance “Path Destination Unknown” Response over TCP

(2) T40 Results

When testing the Instance field, Class is set to 0x01. Requests with Instance 0x00 and 0x01 return “Service not supported” responses (Figure 116). All other fuzzed Instance inputs return “Path destination unknown” messages (Figure 117).

No.	Time	Source	Destination	Protocol	Length	Resp	Info
146	22:13:29.161057	192.168.0.62	192.168.0.59	TCP	60		41892 → 44818 [ACK] Seq=2191 Ack=2145 Win=29200 Len=0
147	22:13:29.218914	192.168.0.62	192.168.0.59	CIP	101		Identity - Find Next Object Instance
148	22:13:29.238297	192.168.0.59	192.168.0.62	CIP	100		Service not supported: Identity - Find Next Object Instance
149	22:13:29.269007	192.168.0.62	192.168.0.59	TCP	60		41892 → 44818 [ACK] Seq=2238 Ack=2191 Win=29200 Len=0
150	22:13:29.324587	192.168.0.62	192.168.0.59	CIP	101		Identity - Find Next Object Instance
151	22:13:29.338469	192.168.0.59	192.168.0.62	CIP	100		Path destination unknown: Identity - Find Next Object Instance
Common Industrial Protocol							
Service: Find Next Object Instance (Response)							
1... = Request/Response: Response (0x1)							
.001 0001 = Service: Find Next Object Instance (0x11)							
Status: Service not supported:							
General Status: Service not supported (0x00)							
Additional Status Size: 1 (words)							
Additional Status							
Additional Status: 0x0000							
[Request Path Size: 2 (words)]							
[Request Path: Identity, Instance: 0x01]							
[Path Segment: 0x20 (8-Bit Class Segment)]							
[001. = Path Segment Type: Logical Segment (1)]							
[...0 00.. = Logical Segment Type: Class ID (0)]							
[.... ..00 = Logical Segment Format: 8-bit Logical Segment (0)]							
[8-Bit Class Segment]							
[Path Segment: 0x24 (8-Bit Instance Segment)]							
[001. = Path Segment Type: Logical Segment (1)]							
[...0 01.. = Logical Segment Type: Instance ID (1)]							
[.... ..00 = Logical Segment Format: 8-bit Logical Segment (0)]							
[8-Bit Instance Segment]							
0000	00 0c 29 56 20 17 00 1d	9c a1 28 4c 00 00 45 00	..)V(L..E.				
0010	00 56 0e 4a 00 00 00 06	aa b8 c0 a0 00 3b c0 a8	.V.				
0020	00 3e af 12 a3 a4 4d 87	b6 91 37 de 82 82 50 18	.>....M. ..7..P.				
0030	07 00 b2 6e 00 00 6f 00	16 00 45 35 45 29 00 000. ..E5E)..				
0040	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00				
0050	00 00 00 04 02 00 00 00	00 00 b2 00 06 00 91 00				
0060	08 01 00 00					

Figure 116. Find_Next_Object_Instance “Service Not Supported” Response over TCP

No.	Time	Source	Destination	Protocol	Length	Resp	Info
21	22:13:24.769308	192.168.0.62	192.168.0.59	CIP	101		Identity - Find Next Object Instance
22	22:13:24.789250	192.168.0.59	192.168.0.62	CIP	100		Path destination unknown: Identity - Find Next Object Instance
23	22:13:24.789910	192.168.0.62	192.168.0.59	TCP	60		41892 → 44818 [ACK] Seq=264 Ack=259 Win=29200 Len=0
24	22:13:24.875272	192.168.0.62	192.168.0.59	CIP	101		Identity - Find Next Object Instance
25	22:13:24.880299	192.168.0.59	192.168.0.62	CIP	100		Path destination unknown: Identity - Find Next Object Instance
26	22:13:24.880747	192.168.0.62	192.168.0.59	TCP	60		41892 → 44818 [ACK] Seq=311 Ack=305 Win=29200 Len=0
Common Industrial Protocol							
Service: Find Next Object Instance (Response)							
1... = Request/Response: Response (0x1)							
.001 0001 = Service: Find Next Object Instance (0x11)							
Status: Path destination unknown:							
General Status: Path destination unknown (0x05)							
Additional Status Size: 1 (words)							
Additional Status							
Additional Status: 0x0000							
[Request Path Size: 2 (words)]							
[Request Path: Identity, Instance: 0x26]							
[Path Segment: 0x20 (8-Bit Class Segment)]							
[001. = Path Segment Type: Logical Segment (1)]							
[...0 00.. = Logical Segment Type: Class ID (0)]							
[.... ..00 = Logical Segment Format: 8-bit Logical Segment (0)]							
[8-Bit Class Segment]							
[Class: Identity (0x01)]							
[Path Segment: 0x24 (8-Bit Instance Segment)]							
[001. = Path Segment Type: Logical Segment (1)]							
[...0 01.. = Logical Segment Type: Instance ID (1)]							
[.... ..00 = Logical Segment Format: 8-bit Logical Segment (0)]							
[8-Bit Instance Segment]							
[Instance: 0x26]							
0000	00 0c 29 56 20 17 00 1d	9c a1 28 4c 00 00 45 00	..)V(L..E.				
0010	00 56 0e 20 00 00 00 06	aa b8 c0 a0 00 3b c0 a8	.V.				
0020	00 3e af 12 a3 a4 4d 87	af 05 37 de 7a cc 50 18	.>....M. ..7..P.				
0030	07 00 c4 b0 00 00 6f 00	16 00 45 35 45 29 00 000. ..E5E)..				
0040	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00				
0050	00 00 00 04 02 00 00 00	00 00 b2 00 06 00 91 00				
0060	05 01 00 00					

Figure 117. Find_Next_Object_Instance “Path Destination Unknown” Response over TCP

(3) T41 Results

The Maximum Returned Values field is tested with inputs between 0x00 and 0xFF. All requests return General Status 0x08 “Service not supported” [15] responses when the Class is set to 0x01 and Instance is set to 0x00 (Figure 118).

No.	Time	Source	Destination	Protocol	Length	Resp	Info
952	22:17:22.449165	192.168.0.62	192.168.0.59	CIP	181		Identity - Find Next Object Instance
953	22:17:22.455947	192.168.0.59	192.168.0.62	CIP	100		Service not supported: Identity - Find Next Object Instance
954	22:17:22.493898	192.168.0.62	192.168.0.59	TCP	60		41894 → 44818 [ACK] Seq=14881 Ack=14565 Win=29200 Len=0
955	22:17:22.554262	192.168.0.62	192.168.0.59	CIP	181		Identity - Find Next Object Instance
956	22:17:22.565952	192.168.0.59	192.168.0.62	CIP	100		Service not supported: Identity - Find Next Object Instance
957	22:17:22.606157	192.168.0.62	192.168.0.59	TCP	60		41894 → 44818 [ACK] Seq=14928 Ack=14611 Win=29200 Len=0

▶ Ethernet/IP (Industrial Protocol), Session: 0x44F898FE, Send RR Data	
▼ Common Industrial Protocol	
▶ Service: Find Next Object Instance (Response)	
▶ Status: Service not supported:	
[Request Path Size: 2 (words)]	
▼ [Request Path: Identity, Instance: 0x00]	
▼ [Path Segment: 0x20 (8-Bit Class Segment)]	
[001. = Path Segment Type: Logical Segment (1)]	
[...0 00.. = Logical Segment Type: Class ID (0)]	
[.... ..00 = Logical Segment Format: 8-bit Logical Segment (0)]	
▼ [8-Bit Class Segment]	
[Class: Identity (0x01)]	
▼ [Path Segment: 0x24 (8-Bit Instance Segment)]	
[001. = Path Segment Type: Logical Segment (1)]	
[...0 01.. = Logical Segment Type: Instance ID (1)]	
[.... ..00 = Logical Segment Format: 8-bit Logical Segment (0)]	
▼ [8-Bit Instance Segment]	
[Instance: 0x00]	

0000	00 0c 29 56 20 17 00 1d 9c a1 28 4c 08 00 45 00	..)V(L..E.
0010	00 56 10 e1 00 00 06 a7 f7 c0 a0 00 3b c0 a0	.V.....
0020	00 3e af 12 a3 a6 49 42 09 0a f1 42 19 49 50 18	..B...IB ...B.IP.
0030	07 d0 1d 0e 00 00 6f 00 16 00 fe 98 fb 44 00 00O.D..
0040	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0050	00 00 00 04 02 00 00 00 00 00 b2 00 06 00 91 00
0060	08 01 00 00

Figure 118. Find_Next_Object_Instance “Service Not Supported” Response over TCP

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C. PCCC COMMAND RESPONSES

The following Wireshark captures in Figures 119 - 154 illustrate test case responses for each command. For descriptions of SUT responses, see Chapter V: Test Analysis.

A. PCCC ECHO TEST CASES

This section shows the results of the PCCC Echo test cases.

(1) T42 Results

No.	Time	Source	Destination	Protocol	Length	Resp	Info
17	10:21:38.771227	192.168.0.62	192.168.0.59	CIP	118		Class (0x67) - Service (0x4b)
18	10:21:38.778627	192.168.0.59	192.168.0.62	CIP	115		Success: Class (0x67) - Service (0x4b)
19	10:21:38.784119	192.168.0.62	192.168.0.59	CIP	118		Class (0x67) - Service (0x4b)
20	10:21:38.798564	192.168.0.59	192.168.0.62	CIP	115		Success: Class (0x67) - Service (0x4b)
21	10:21:38.807709	192.168.0.62	192.168.0.59	CIP	118		Class (0x67) - Service (0x4b)

▶ Frame 18: 115 bytes on wire (920 bits), 115 bytes captured (920 bits) on interface 0 ▶ Ethernet II, Src: Rockwell_a1:28:4c (00:1d:9c:a1:28:4c), Dst: Vmware_56:20:17 (00:0c:29:56:20:17) ▶ Internet Protocol Version 4, Src: 192.168.0.59, Dst: 192.168.0.62 ▶ Transmission Control Protocol, Src Port: 44818, Dst Port: 54640, Seq: 160, Ack: 243, Len: 61 ▶ EtherNet/IP (Industrial Protocol), Session: 0xD05A2C79, Send Unit Data ▶ Common Industrial Protocol ▼ CIP Class Generic ▼ Command Specific Data Data: 074d00f30a600546000100							
0000	00	0c	29	56	20	17	00 1d 9c a1 28 4c 00 00 45 00 ..)V(L..E.
0010	00	65	00	05	00	00	00 06 b8 c4 c0 a8 00 3b c0 a8 .e.....;..
0020	00	3e	af	12	d5	70	49 0b 5d c7 6b e0 c7 ae 50 18 .>...pI.].k...P.
0030	07	d0	18	a9	00	00	70 00 25 00 79 2c 5a d0 00 00p. %.y,Z...
0040	00	00	00	00	00	00	00 00 00 00 00 00 00 00 00 0050 00 00 00 00 02 00 a1 00 04 00 01 00 fe 80 b1 00 0060 11 00 82 00 cb 00 00 00 07 4d 00 f3 0a 60 05 46M...F 0070 00 01 00 ...

Figure 119. Echo Response over TCP (0 Bytes Attached)

(2) T43 Results

No.	Time	Source	Destination	Protocol	Length	Resp	Info
✓ 560	12:47:24.188161	192.168.0.59	192.168.0.62	CIP	362		Success: Class (0x67) - Service (0x4b)
✓ 561	12:47:24.198649	192.168.0.62	192.168.0.59	CIP	365		Class (0x67) - Service (0x4b)
✓ 562	12:47:24.208645	192.168.0.59	192.168.0.62	CIP	362		Success: Class (0x67) - Service (0x4b)
✓ 563	12:47:24.218275	192.168.0.62	192.168.0.59	CIP	365		Class (0x67) - Service (0x4b)
✓ 564	12:47:24.228650	192.168.0.59	192.168.0.62	CIP	362		Success: Class (0x67) - Service (0x4b)
▶ Frame 563: 365 bytes on wire (2920 bits), 365 bytes captured (2920 bits) on interface 0 ▶ Ethernet II, Src: Vmware_56:20:17 (00:0c:29:56:20:17), Dst: Rockwell_a1:28:4c (00:1d:9c:a1:28:4c) ▶ Internet Protocol Version 4, Src: 192.168.0.62, Dst: 192.168.0.59 ▶ Transmission Control Protocol, Src Port: 54842, Dst Port: 44818, Seq: 15354, Ack: 15191, Len: 311 ▶ EtherNet/IP (Industrial Protocol), Session: 0x41DFADF6, Send Unit Data ▶ Common Industrial Protocol ▼ CIP Class Generic ▼ Command Specific Data Data: 074d00f30a60506000100008888888888888888888888888888...							
0000	00 1d 9c a1 28 4c 00 0c	29 56 20 17 00 00 45 00(L..)V...E.				
0010	01 5f cd 0d 40 00 40 06	ea c1 c0 a8 00 3e c0 a8	...@.>..				
0020	00 3b d6 3a af 12 e9 5f	b5 92 4d 5f 0b d0 50 18	.;.....M..P.				
0030	ff 70 d4 88 00 00 70 00	1f 01 f6 ad df 41 00 00	.p....p.A.				
0040	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00				
0050	00 00 14 00 02 00 a1 00	04 00 d6 cb ad f6 b1 00				
0060	0b 01 32 00 4b 02 20 67	24 01 07 4d 00 f3 0a 60	..2.K. g S..M..				
0070	05 06 00 01 00 00 00 00	08 08 08 08 08 08 08 08				
0080	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00				
0090	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00				
00a0	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00				
00b0	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00				
00c0	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00				
00d0	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00				
00e0	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00				
00f0	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00				
0100	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00				
0110	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00				
0120	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00				
0130	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00				
0140	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00				
0150	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00				
0160	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00				

Figure 120. Echo Response over TCP (243 Bytes Attached)

(3) T44 Results

No.	Time	Source	Destination	Protocol	Length	Resp	Info
✓ 9	18:03:28.772176	192.168.0.62	192.168.0.59	CIP	126		Class (0x67) - Service (0x4b)
✓ 10	18:03:28.776256	192.168.0.59	192.168.0.62	CIP	123		Success: Class (0x67) - Service (0x4b)
✓ 11	18:03:28.812534	192.168.0.62	192.168.0.59	CIP	126		Class (0x67) - Service (0x4b)
✓ 12	18:03:28.826226	192.168.0.59	192.168.0.62	CIP	123		Success: Class (0x67) - Service (0x4b)
✓ 13	18:03:28.865554	192.168.0.62	192.168.0.59	CIP	126		Class (0x67) - Service (0x4b)
▶ Ethernet II, Src: Rockwell_a1:28:4c (00:1d:9c:a1:28:4c), Dst: Vmware_56:20:17 (00:0c:29:56:20:17) ▶ Internet Protocol Version 4, Src: 192.168.0.59, Dst: 192.168.0.62 ▶ Transmission Control Protocol, Src Port: 44818, Dst Port: 41950, Seq: 99, Ack: 187, Len: 69 ▶ EtherNet/IP (Industrial Protocol), Session: 0x23431403, Send Unit Data ▶ Common Industrial Protocol ▼ CIP Class Generic ▼ Command Specific Data Data: 074d00f30a600546000100e3c605320e99a816							
0000	00 0c 29 56 20 17 00 1d	9c a1 28 4c 00 00 45 00	..)V(L..E.				
0010	00 6d 00 a5 00 00 00 06	b8 1c c0 a8 00 3b c0 a8	.m.....:..				
0020	00 3e af 12 a3 de 4d 54	94 f5 c5 c0 97 d8 50 18	.>....MTP.				
0030	07 d0 e2 ce 00 00 70 00	2d 00 d3 14 43 23 00 00p.C#..				
0040	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00				
0050	00 00 00 00 02 00 a1 00	04 00 01 00 fe 00 b1 00				
0060	19 00 01 00 cb 00 00 00	07 4d 00 f3 0a 60 05 46M.....F				
0070	00 01 00 e3 c6 05 32 0e	99 a8 162. ...				

Figure 121. Echo Response over TCP (8 Bytes Fuzzed)

(4) T45 Results

No.	Time	Source	Destination	Protocol	Length	Resp	Info
1047	18:05:50.430294	192.168.0.62	192.168.0.59	CIP	127		Class (0x67) - Service (0x4b)
1048	18:05:50.443166	192.168.0.59	192.168.0.62	CIP	124		Success: Class (0x67) - Service (0x4b)
1049	18:05:50.477308	192.168.0.62	192.168.0.59	CIP	127		Class (0x67) - Service (0x4b)
1050	18:05:50.483309	192.168.0.59	192.168.0.62	CIP	124		Success: Class (0x67) - Service (0x4b)
1051	18:05:50.520187	192.168.0.62	192.168.0.59	CIP	127		Class (0x67) - Service (0x4b)
▶ Ethernet II, Src: Rockwell_a1:28:4c (00:1d:9c:a1:28:4c), Dst: Vmware_56:20:17 (00:0c:29:56:20:17) ▶ Internet Protocol Version 4, Src: 192.168.0.59, Dst: 192.168.0.62 ▶ Transmission Control Protocol, Src Port: 44818, Dst Port: 41952, Seq: 32299, Ack: 33768, Len: 70 ▶ Ethernet/IP (Industrial Protocol), Session: 0xA3DB211D, Send Unit Data ▶ Common Industrial Protocol ▼ CIP Class Generic ▼ Command Specific Data Data: 074d00f30a6005460001002144721d32c0204feb							
0000	00 0c 29 56 20 17 00 1d	9c a1 28 4c 08 00 45 00	..)V(L..E.				
0010	00 6e 07 09 00 00 00 06	b0 e7 c0 a8 00 3b c0 a8	.n.....				
0020	00 3e af 12 a3 e0 49 0e	52 35 0f d6 53 30 50 18	.>....T. R5..50P.				
0030	07 d0 aa ac 00 00 70 00	2e 00 1d 21 db a3 00 00p.!				
0040	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00				
0050	00 00 00 00 02 00 a1 00	04 00 01 00 fe 80 b1 00				
0060	1a 00 cd 01 cb 00 00 00	07 4d 00 f3 0a 60 05 46M....'.F				
0070	00 01 00 21 44 72 1d 32	c0 20 4f eb	...!Dr.2 . 0.				

Figure 122. Echo Response over TCP (9 Bytes Fuzzed)

(5) T46 Results

No.	Time	Source	Destination	Protocol	Length	Resp	Info
4858	13:06:15.393961	192.168.0.62	192.168.0.59	CIP	128		Class (0x67) - Service (0x4b)
4859	13:06:15.396957	192.168.0.59	192.168.0.62	CIP	125		Success: Class (0x67) - Service (0x4b)
4860	13:06:15.430753	192.168.0.62	192.168.0.59	CIP	128		Class (0x67) - Service (0x4b)
4861	13:06:15.436653	192.168.0.59	192.168.0.62	CIP	125		Success: Class (0x67) - Service (0x4b)
4862	13:06:15.476375	192.168.0.62	192.168.0.59	TCP	60		54846 → 44818 [ACK] Seq=160399 Ack=153885 Win=29200 Len=0
▶ Frame 4859: 125 bytes on wire (1000 bits), 125 bytes captured (1000 bits) on interface 0 ▶ Ethernet II, Src: Rockwell_a1:28:4c (00:1d:9c:a1:28:4c), Dst: Vmware_56:20:17 (00:0c:29:56:20:17) ▶ Internet Protocol Version 4, Src: 192.168.0.59, Dst: 192.168.0.62 ▶ Transmission Control Protocol, Src Port: 44818, Dst Port: 54846, Seq: 153743, Ack: 160325, Len: 71 ▶ Ethernet/IP (Industrial Protocol), Session: 0xBEABC99A, Send Unit Data ▶ Common Industrial Protocol ▶ CIP Class Generic							
0000	00 0c 29 56 20 17 00 1d	9c a1 28 4c 08 00 45 00	..)V(L..E.				
0010	00 6f 34 01 00 00 00 06	84 be c0 a8 00 3b c0 a8	.o4.....				
0020	00 3e af 12 d6 3e 4d 63	59 56 8d 7c 32 53 50 18	.>....Mc YV.[25P.				
0030	07 d0 aa 5e 00 00 70 00	2f 00 9a c9 a8 be 00 00	...^..p. /.....				
0040	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00				
0050	00 00 00 00 02 00 a1 00	04 00 01 00 fe 80 b1 00				
0060	1b 00 75 08 cb 00 00 00	07 4d 00 f3 0a 60 05 46	..U......M....'.F				
0070	00 01 00 00 5c 83 eb cb	b5 2f e4 67 00\.... ./..g.				

Figure 123. Echo Response over TCP (10 Bytes Fuzzed)

(6) T47 Results

No.	Time	Source	Destination	Protocol	Length	Resp	Info
4920	13:34:58.830317	192.168.0.62	192.168.0.59	CIP	158		Class (0x67) - Service (0x4b)
4921	13:34:58.844916	192.168.0.59	192.168.0.62	CIP	155		Success: Class (0x67) - Service (0x4b)
4922	13:34:58.846818	192.168.0.62	192.168.0.59	TCP	60		54854 → 44818 [ACK] Seq=170155 Ack=165234 Win=29200 Len=0
4923	13:34:58.969471	192.168.0.62	192.168.0.59	CIP	158		Class (0x67) - Service (0x4b)
4924	13:34:58.983985	192.168.0.59	192.168.0.62	CIP	155		Success: Class (0x67) - Service (0x4b)
▶ Ethernet II, Src: Rockwell_a1:28:4c (00:1d:9c:a1:28:4c), Dst: Vmware_56:20:17 (00:0c:29:56:20:17)							
▶ Internet Protocol Version 4, Src: 192.168.0.59, Dst: 192.168.0.62							
▶ Transmission Control Protocol, Src Port: 44818, Dst Port: 54854, Seq: 165133, Ack: 170155, Len: 101							
▶ EtherNet/IP (Industrial Protocol), Session: 0x34C2C674, Send Unit Data							
▶ Common Industrial Protocol							
▼ CIP Class Generic							
▼ Command Specific Data							
Data: 074d00f30a600546000100c1505c1c88c58e1d7e25aa285...							
0000	00 0c 29 56 20 17 00 1d	9c a1 28 4c 00 00 45 00	..)V(L..E.				
0010	00 0d 5b 31 00 00 00 06	5d 70 c0 a0 00 3b c0 a0	..[1....]p...;..				
0020	00 3e af 12 d6 46 4d 6f	82 7d 67 77 31 a1 50 18	>...FMO .jgw1.P.				
0030	07 d0 71 30 00 00 70 00	4d 00 74 c6 c2 34 00 00	..Q0..p. M.t...4..				
0040	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00				
0050	00 00 00 00 02 00 a1 00	04 00 01 00 fe 00 b1 00				
0060	39 00 63 06 cb 00 00 00	07 4d 00 f3 0a 60 05 46	9.C..... .M...'.F				
0070	00 01 00 c1 50 5c 1c 88	8c 58 e1 d7 e2 5a a2 85P\... .X...Z..				
0080	fc 46 18 0b 66 80 df 88	6b 48 25 12 91 cb c7 76	.F..f... kWh....v				
0090	5f b3 14 c1 96 2f 1a 61	32 1e 00/.a 2..				

Figure 124. Echo Response over TCP (40 Bytes Fuzzed)

(7) T48 Results

No.	Time	Source	Destination	Protocol	Length	Resp	Info
14	16:09:05.290622	192.168.0.59	192.168.0.62	CIP	362		Success: Class (0x67) - Service (0x4b)
15	16:09:05.290794	192.168.0.62	192.168.0.59	TCP	60		60858 → 44818 [ACK] Seq=737 Ack=715 Win=31088 Len=0
16	16:09:08.027479	192.168.0.62	192.168.0.59	CIP	365		Class (0x67) - Service (0x4b)
17	16:09:08.040732	192.168.0.59	192.168.0.62	CIP	362		Success: Class (0x67) - Service (0x4b)
18	16:09:08.041120	192.168.0.62	192.168.0.59	TCP	60		60858 → 44818 [ACK] Seq=1048 Ack=1023 Win=32160 Len=0
19	16:09:10.693445	192.168.0.62	192.168.0.59	CIP	365		Class (0x67) - Service (0x4b)
20	16:09:10.700684	192.168.0.59	192.168.0.62	CIP	362		Success: Class (0x67) - Service (0x4b)
▶ Frame 17: 362 bytes on wire (2896 bits), 362 bytes captured (2896 bits) on interface 0							
▶ Ethernet II, Src: Rockwell_a1:28:4c (00:1d:9c:a1:28:4c), Dst: Vmware_56:20:17 (00:0c:29:56:20:17)							
▶ Internet Protocol Version 4, Src: 192.168.0.59, Dst: 192.168.0.62							
▶ Transmission Control Protocol, Src Port: 44818, Dst Port: 60858, Seq: 715, Ack: 1048, Len: 308							
▶ EtherNet/IP (Industrial Protocol), Session: 0xABAS0541, Send Unit Data							
▶ Common Industrial Protocol							
▼ CIP Class Generic							
▼ Command Specific Data							
Data: 074d00f30a60054600010034e89d3a22ef738ef6953073fe...							
0000	00 0c 29 56 20 17 00 1d	9c a1 28 4c 00 00 45 00	..)V(L..E.				
0010	01 5c 7d 2c 00 00 00 06	3a a6 c0 a0 00 3b c0 a0	..),... :...;..				
0020	00 3e af 12 ed ba 49 2e	10 b1 33 b2 8f 9f 50 18	>....I. ..3...P.				
0030	07 d0 8a a7 00 00 70 00	1c 01 41 05 a5 ab 00 00p. ..A.....				
0040	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00				
0050	00 00 00 00 02 00 a1 00	04 00 01 00 fe 00 b1 00				
0060	00 01 03 00 cb 00 00 00	07 4d 00 f3 0a 60 05 46M...'.F				
0070	00 01 00 34 e8 9d 3a 22	ef 73 8e f6 95 30 73 fe	...4...". s...0s.				
0080	2c 3b 13 f2 54 72 7f 40	f2 7a 95 c6 2e 64 85 64	...Tr.@ .z...d.d				
0090	c8 7d 19 a7 a3 26 d8 22	d9 fa d9 35 6e 7c 26 fa	...&.. " ...5n &				
00a0	e8 96 0a 4b 9a 6d cb 14	e3 b6 1e d4 6f ab e4 d9	...K.m.O...				
00b0	f6 30 70 6b a9 3a 53 4f	6f 48 73 15 39 18 1c 9b	.0pk.:50 oHs.9...				
00c0	b2 70 2a a8 4f 93 31 43	d9 d4 96 50 92 45 9a c8	.pw.0.1C ...P.E..				
00d0	dd da 71 99 a8 2e 2a d1	78 13 14 27 4a ef 84 8d	..q...*. X..'.j...				
00e0	6d f6 c3 ec 00 5e 9a ab	d1 ca dd 78 44 49 65 77	m...^... ..xDIEW				
00f0	97 56 94 74 72 9c a2 95	cb 09 19 46 19 27 ca bd	.V.tr... ..F...'.				
0100	48 dc ab b8 89 67 df ea	7b 0f 39 69 ba 4c 83 68	H....g.. [.9i.L.h				
0110	4c a9 b4 10 9a 0b c6 36	5d f8 39 01 67 43 06 69	L.....6]0.g.C.i				
0120	25 7b 8f 0e 64 d2 20 f8	e1 a8 78 a1 6a f4 19 99	%{.d. . .X.j...b				
0130	00 97 33 e9 6d 9f 98 c0	2a 95 81 98 da d0 e8 62	..3.m... *......b				
0140	dc 4b e2 2e 3d 27 f0 1b	7e 63 ff 60 d4 2e 4d e2	.K..='.. ~c..'.M.				
0150	d0 d3 00 1e 78 68 43 7d	13 61 71 fe e6 5d d4 e0xhC[. ~aq...].				
0160	8a 01 39 43 1f 15 4b e2	b3 13	..9C..K. ..				

Figure 125. Echo Response over TCP (243 Bytes Fuzzed)

(8) T49 Results

No.	Time	Source	Destination	Protocol	Length	Resp	Info
9	09:16:15.053937	192.168.0.62	192.168.0.59	TCP	60		40348 → 44818 [ACK] Seq=115 Ack=99 Win=29200 Len=0
10	09:16:17.751022	192.168.0.62	192.168.0.59	CIP	365		Class (0x67) - Service (0x4b)
11	09:16:17.761853	192.168.0.59	192.168.0.62	CIP	362		Success: Class (0x67) - Service (0x4b)
12	09:16:17.762238	192.168.0.62	192.168.0.59	TCP	60		40348 → 44818 [ACK] Seq=426 Ack=407 Win=30016 Len=0
13	09:16:20.554558	192.168.0.62	192.168.0.59	CIP	365		Class (0x67) - Service (0x4b)
▶ Ethernet II, Src: Rockwell_a1:28:4c (00:1d:9c:a1:28:4c), Dst: Vmware_56:20:17 (00:0c:29:56:20:17) ▶ Internet Protocol Version 4, Src: 192.168.0.59, Dst: 192.168.0.62 ▶ Transmission Control Protocol, Src Port: 44818, Dst Port: 40348, Seq: 99, Ack: 426, Len: 308 ▶ Ethernet/IP (Industrial Protocol), Session: 0x0FFD7F87, Send Unit Data ▶ Common Industrial Protocol ▼ CIP Class Generic ▼ Command Specific Data Data: 074d00f30a600546000100b696cd746d1b7e5c66f91a1df2...							
0000	00 0c 29 56 20 17 00 1d	9c a1 28 4c 08 00 45 00	..)V(L..E.				
0010	01 5c 00 19 00 00 00 06	b7 b9 c0 a8 00 3b c0 a8	.\.....				
0020	00 3e af 12 9d 9c 4d 53	0d 78 7a 34 aa 03 50 18	.>....MS .xz4.P.				
0030	07 d0 40 4a 00 00 70 00	1c 01 87 7f fd 0f 00 00	..@J..P.				
0040	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00				
0050	00 00 00 00 02 00 a1 00	04 00 01 00 fe 08 b1 00				
0060	00 01 01 00 cb 00 00 00	07 4d 00 f3 0a 60 05 46M...'.F				
0070	00 01 00 b6 96 cd 74 6d	1b 7e 5c 66 f9 1a 1d f2tm ~\f....				
0080	f1 f4 db 85 1f 08 10 d4	d1 fc 09 73 55 5a bb c4sUZ..				
0090	50 4b 05 2d ba 5d b5 53	ee 56 96 66 1b 59 88 0c	PK.-.].S .V.f.Y..				
00a0	67 76 5c 67 3a 36 b6 7e	e8 5e f4 72 6c 04 4c 3e	gv\g:6k~ .^..r.L>				
00b0	47 21 8f c2 41 a5 e8 51	67 10 94 4d 98 5f cc c3	G!..A..Q g..M..._				
00c0	20 ef 0b f5 dc 17 d2 21	44 1a 79 a8 c7 9e 6b 32! Dey...k2				
00d0	0f 85 4a d5 77 d5 d9 af	0d 09 08 c6 45 64 f4 3b	..J.W... ..Ed.;				
00e0	67 9c 1f d6 23 49 f5 55	48 11 70 46 94 83 0f 1e	g...#I.U H.pF....				
00f0	8a 29 97 e0 32 36 26 bf	75 d7 4e d1 b2 af 51 6e	.).26&. u.N...0n				
0100	29 5b a1 d5 a7 b0 a2 bc	78 52 e0 2b 5b 6c 8b 86)[..... xR.+[L.				
0110	e9 da fd 6f f7 37 88 10	e9 2f f0 4d bd 62 bf 77	...o.7.. ./M.b.w				
0120	e2 97 f7 70 d2 ac 25 0e	8e 6d 51 5c ff 77 d5 5e	...p.%..mQ\w.^				
0130	e4 01 0e b0 4d 08 8c fd	0d 1f 8e 73 b0 89 ec e4M... ..S....				
0140	4f 43 00 15 08 79 4b df	2f 88 96 a3 1f 8e 0e 14	0C...yK. /...V...9				
0150	de 4f 2b d4 16 79 32 c6	92 ab b8 56 9e 1d 09 39	.0+..y2. ...V...9				
0160	98 1c 4e 75 67 3c ae 0e	db a9 b8 56 9e 1d 09 39	..Nug<... ..				

Figure 126. Echo Response over TCP (Maximum Number of Bytes without Error Message: 247 Bytes Fuzzed)

(9) T50 Results

No.	Time	Source	Destination	Protocol	Length	Resp	Info
674	12:47:38.884292	192.168.0.62	192.168.0.59	CIP	366		Class (0x67) - Service (0x4b)
675	12:47:38.887580	192.168.0.59	192.168.0.62	CIP	184		Routing failure, request packet too large: Class (0x67) - Service (0x4b)
676	12:47:38.899296	192.168.0.62	192.168.0.59	CIP	366		Class (0x67) - Service (0x4b)
677	12:47:38.907682	192.168.0.59	192.168.0.62	CIP	184		Routing failure, request packet too large: Class (0x67) - Service (0x4b)
▶ Ethernet/IP (Industrial Protocol), Session: 0x473EBFB9, Send Unit Data ▼ Common Industrial Protocol ▼ Service: Unknown Service (0x4b) (Response) 1... .. = Request/Response: Response (0x1) .100 1011 = Service: Unknown (0x4b) ▼ Status: Routing failure, request packet too large: General Status: Routing failure, request packet too large (0x1a) Additional Status Size: 0 (words) [Request Path Size: 2 (words)] ▼ [Request Path: Class: 0x67, Instance: 0x01] ▼ [Path Segment: 0x20 (8-Bit Class Segment)] [001. = Path Segment Type: Logical Segment (1)] [...0 00.. = Logical Segment Type: Class ID (0)] [.... ..00 = Logical Segment Format: 8-bit Logical Segment (0)] ▼ [8-Bit Class Segment] [Class: Unknown (0x67)] ▼ [Path Segment: 0x24 (8-Bit Instance Segment)] [001. = Path Segment Type: Logical Segment (1)] [...0 01.. = Logical Segment Type: Instance ID (1)] [.... ..00 = Logical Segment Format: 8-bit Logical Segment (0)] ▼ [8-Bit Instance Segment] [Instance: 0x01]							
0000	00 0c 29 56 20 17 00 1d	9c a1 28 4c 08 00 45 00	..)V(L..E.				
0010	00 5a 2b 06 00 00 00 06	0d 4e c0 a8 00 3b c0 a8	.Z+.....N...;..				
0020	00 3e af 12 d6 3c 49 22	0c 44 ac 70 f0 fb 50 18	.>...<I" .D.p..P.				
0030	07 d0 b7 53 00 00 70 00	1a 00 b9 bf 3e 47 00 00	...S..P.>G..				
0040	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00				
0050	00 00 00 02 00 a1 00	04 00 01 00 fe 08 b1 00				
0060	06 00 32 00 cb 00 1a 00		..Z.....				

Figure 127. Echo Response over TCP (248 Bytes Fuzzed)

(10) T51 Results

No.	Time	Source	Destination	Protocol	Length	Resp	Info
22	14:36:15.544165	192.168.0.59	192.168.0.62	CIP	104		Routing failure, request packet too large: Class (0x67) - Service (0x4b)
23	14:36:15.553019	192.168.0.62	192.168.0.59	CIP	374		Class (0x67) - Service (0x4b)
24	14:36:15.564003	192.168.0.59	192.168.0.62	CIP	104		Routing failure, request packet too large: Class (0x67) - Service (0x4b)
25	14:36:15.575199	192.168.0.62	192.168.0.59	CIP	374		Class (0x67) - Service (0x4b)
26	14:36:15.584990	192.168.0.59	192.168.0.62	CIP	104		Routing failure, request packet too large: Class (0x67) - Service (0x4b)
▶ Transmission Control Protocol, Src Port: 44818, Dst Port: 59446, Seq: 449, Ack: 2675, Len: 50							
▶ Ethernet/IP (Industrial Protocol), Session: 0x41D08565, Send Unit Data							
▼ Common Industrial Protocol							
▼ Service: Unknown Service (0x4b) (Response)							
1... = Request/Response: Response (0x1)							
.100 1011 = Service: Unknown (0x4b)							
▼ Status: Routing failure, request packet too large:							
General Status: Routing failure, request packet too large (0x1a)							
Additional Status Size: 0 (words)							
[Request Path Size: 2 (words)]							
▼ [Request Path: Class: 0x67, Instance: 0x01]							
▼ [Path Segment: 0x20 (8-Bit Class Segment)]							
[001. = Path Segment Type: Logical Segment (1)]							
[...0 00.. = Logical Segment Type: Class ID (0)]							
[.... ..00 = Logical Segment Format: 8-bit Logical Segment (0)]							
▼ [8-Bit Class Segment]							
[Class: Unknown (0x67)]							
▼ [Path Segment: 0x24 (8-Bit Instance Segment)]							
[001. = Path Segment Type: Logical Segment (1)]							
[...0 01.. = Logical Segment Type: Instance ID (1)]							
[.... ..00 = Logical Segment Format: 8-bit Logical Segment (0)]							
▼ [8-Bit Instance Segment]							
[Instance: 0x01]							
0000	00 0c 29 56 20 17 00 1d	9c a1 28 4c 08 00 45 00	..)V(L..E.				
0010	00 5a 09 21 00 00 00 06	af b3 c0 a8 00 3b c0 a8	.Z.!.../..				
0020	00 3e af 12 e8 36 49 1a	5e 71 58 33 f9 01 50 18	.>...6I. ^qX3..P.				
0030	07 d0 00 ab 00 00 70 00	1a 00 65 85 db a1 00 00P. ..e..A..				
0040	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00				
0050	00 00 00 00 02 00 a1 00	04 00 01 00 fe 80 b1 00				
0060	06 00 00 00 cb 00 1a 00					

Figure 128. Echo Response over TCP (256 Bytes Fuzzed)

B. PCCC PROTECTED TYPED FILE READ TEST CASES

This section shows the results of the PCCC Protected Typed File Read test cases.

(1) T52 Results

No.	Time	Source	Destination	Protocol	Length	Resp	Info
14	14:46:50.062026	192.168.0.59	192.168.0.62	CIP	115		Success: Class (0x67) - Service (0x4b)
15	14:46:50.072149	192.168.0.62	192.168.0.59	CIP	124		Class (0x67) - Service (0x4b)
16	14:46:50.081258	192.168.0.59	192.168.0.62	CIP	115		Success: Class (0x67) - Service (0x4b)
17	14:46:50.093189	192.168.0.62	192.168.0.59	CIP	124		Class (0x67) - Service (0x4b)
18	14:46:50.101030	192.168.0.59	192.168.0.62	CIP	115		Success: Class (0x67) - Service (0x4b)
19	14:46:50.111750	192.168.0.62	192.168.0.59	CIP	124		Class (0x67) - Service (0x4b)
▶ Frame 16: 115 bytes on wire (920 bits), 115 bytes captured (920 bits) on interface 0							
▶ Ethernet II, Src: Rockwell_La1:28:4c (00:1d:9c:a1:28:4c), Dst: Vmware_56:20:17 (00:0c:29:56:20:17)							
▶ Internet Protocol Version 4, Src: 192.168.0.59, Dst: 192.168.0.62							
▶ Transmission Control Protocol, Src Port: 44818, Dst Port: 60268, Seq: 282, Ack: 397, Len: 61							
▶ Ethernet/IP (Industrial Protocol), Session: 0x808C6BC9, Send Unit Data							
▼ Common Industrial Protocol							
▼ CIP Class Generic							
▼ Command Specific Data							
Data: 074d00f30a60054f100200							
0000	00 0c 29 56 20 17 00 1d	9c a1 28 4c 08 00 45 00	..)V(L..E.				
0010	00 65 3c 73 00 00 00 06	7c 56 c0 a8 00 3b c0 a8	.e<.... [V...;..				
0020	00 3e af 12 eb 6c 49 1a	be 56 bc 1b 2f 1b 50 18	.>...lI. .V.../..P.				
0030	07 d0 1e 35 00 00 70 00	25 00 c9 6b bc b8 00 00	...5..p. %.k....				
0040	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00				
0050	00 00 00 00 02 00 a1 00	04 00 01 00 fe 80 b1 00				
0060	11 00 04 00 cb 00 00 00	07 4d 00 f3 0a 60 05 4fM...'.0				
0070	10 02 00		...				

Figure 129. Protected Typed File Read Response over TCP (Size Fuzzed)

(2) T53 Results

No.	Time	Source	Destination	Protocol	Length	Resp	Info
14	15:11:01.903989	192.168.0.59	192.168.0.62	CIP	115		Success: Class (0x67) - Service (0x4b)
15	15:11:01.910966	192.168.0.62	192.168.0.59	CIP	124		Class (0x67) - Service (0x4b)
16	15:11:01.923859	192.168.0.59	192.168.0.62	CIP	115		Success: Class (0x67) - Service (0x4b)
17	15:11:01.933418	192.168.0.62	192.168.0.59	CIP	124		Class (0x67) - Service (0x4b)
18	15:11:01.943803	192.168.0.59	192.168.0.62	CIP	115		Success: Class (0x67) - Service (0x4b)
19	15:11:01.953638	192.168.0.62	192.168.0.59	CIP	124		Class (0x67) - Service (0x4b)
▶ Frame 16: 115 bytes on wire (920 bits), 115 bytes captured (920 bits) on interface 0 ▶ Ethernet II, Src: Rockwell_a1:28:4c (00:1d:9c:a1:28:4c), Dst: Vmware_56:20:17 (00:0c:29:56:20:17) ▶ Internet Protocol Version 4, Src: 192.168.0.59, Dst: 192.168.0.62 ▶ Transmission Control Protocol, Src Port: 44818, Dst Port: 60270, Seq: 221, Ack: 327, Len: 61 ▶ EtherNet/IP (Industrial Protocol), Session: 0xE18F98CD, Send Unit Data ▶ Common Industrial Protocol ▼ CIP Class Generic ▼ Command Specific Data Data: 074d00f30a60054f100200							
0000	00 0c 29 56 20 17 00 1d	9c a1 28 4c 00 00 45 00	..)V(L..E.				
0010	00 65 3f 24 00 00 00 06	79 a5 c0 a8 00 3b c0 a8	.e7\$. ... y.				
0020	00 3e af 12 eb 6e 4d 64	fe 85 c0 48 d6 bf 50 18	.>...nMd ...H..P.				
0030	07 d0 57 8f 00 00 70 00	25 00 cd 9b 8f e1 00 00	..W...p. %.....				
0040	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00				
0050	00 00 00 00 02 00 a1 00	04 00 01 00 fe 80 b1 00				
0060	11 00 03 00 cb 00 00 00	07 4d 00 f3 0a 60 05 4fM...'.0				
0070	10 02 00		...				

Figure 130. Protected Typed File Read Response over TCP (Tag Fuzzed)

(3) T54 Results

No.	Time	Source	Destination	Protocol	Length	Resp	Info
18	15:41:44.318377	192.168.0.59	192.168.0.62	CIP	115		Success: Class (0x67) - Service (0x4b)
19	15:41:44.324552	192.168.0.62	192.168.0.59	CIP	124		Class (0x67) - Service (0x4b)
20	15:41:44.328310	192.168.0.59	192.168.0.62	CIP	115		Success: Class (0x67) - Service (0x4b)
21	15:41:44.338684	192.168.0.62	192.168.0.59	CIP	124		Class (0x67) - Service (0x4b)
22	15:41:44.348393	192.168.0.59	192.168.0.62	CIP	115		Success: Class (0x67) - Service (0x4b)
23	15:41:44.359519	192.168.0.62	192.168.0.59	CIP	124		Class (0x67) - Service (0x4b)
▶ Frame 20: 115 bytes on wire (920 bits), 115 bytes captured (920 bits) on interface 0 ▶ Ethernet II, Src: Rockwell_a1:28:4c (00:1d:9c:a1:28:4c), Dst: Vmware_56:20:17 (00:0c:29:56:20:17) ▶ Internet Protocol Version 4, Src: 192.168.0.59, Dst: 192.168.0.62 ▶ Transmission Control Protocol, Src Port: 44818, Dst Port: 60282, Seq: 484, Ack: 537, Len: 61 ▶ EtherNet/IP (Industrial Protocol), Session: 0x9C72A3AA, Send Unit Data ▶ Common Industrial Protocol ▼ CIP Class Generic							
0000	00 0c 29 56 20 17 00 1d	9c a1 28 4c 00 00 45 00	..)V(L..E.				
0010	00 65 71 6a 00 00 00 06	47 5f c0 a8 00 3b c0 a8	.eqj... G_...;				
0020	00 3e af 12 eb 7a 4d 80	6b 1f 9d 4e 15 87 50 18	.>...zM. k..N..P.				
0030	07 d0 0c 3e 00 00 70 00	25 00 aa a3 72 9c 00 00	...>..p. %...f...				
0040	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00				
0050	00 00 00 00 02 00 a1 00	04 00 01 00 fe 80 b1 00				
0060	11 00 06 00 cb 00 00 00	07 4d 00 f3 0a 60 05 4fM...'.0				
0070	10 02 00		...				

Figure 131. Protected Typed File Read Response over TCP (Offset Fuzzed)

(4) T55 Results

No.	Time	Source	Destination	Protocol	Length	Resp	Info
14	15:39:40.488859	192.168.0.59	192.168.0.62	CIP	115		Success: Class (0x67) - Service (0x4b)
15	15:39:40.498563	192.168.0.62	192.168.0.59	CIP	124		Class (0x67) - Service (0x4b)
16	15:39:40.501720	192.168.0.59	192.168.0.62	CIP	115		Success: Class (0x67) - Service (0x4b)
17	15:39:40.506393	192.168.0.62	192.168.0.59	CIP	124		Class (0x67) - Service (0x4b)
18	15:39:40.511088	192.168.0.59	192.168.0.62	CIP	115		Success: Class (0x67) - Service (0x4b)
19	15:39:40.523738	192.168.0.62	192.168.0.59	CIP	124		Class (0x67) - Service (0x4b)
▶ Frame 16: 115 bytes on wire (920 bits), 115 bytes captured (920 bits) on interface 0 ▶ Ethernet II, Src: Rockwell_a1:28:4c (00:1d:9c:a1:28:4c), Dst: Vmware_56:20:17 (00:0c:29:56:20:17) ▶ Internet Protocol Version 4, Src: 192.168.0.59, Dst: 192.168.0.62 ▶ Transmission Control Protocol, Src Port: 44818, Dst Port: 60280, Seq: 282, Ack: 397, Len: 61 ▶ EtherNet/IP (Industrial Protocol), Session: 0xCA65F4E, Send Unit Data ▶ Common Industrial Protocol ▼ CIP Class Generic ▼ Command Specific Data Data: 074d00f30a60054f100200							
0000	00 0c 29 56 20 17 00 1d 9c a1 28 4c 00 00 45 00	..)V(L..E.					
0010	00 65 65 5b 00 00 00 06 53 6e c0 a0 00 3b c0 a8	.ee[.... Sn...:..					
0020	00 3e af 12 eb 78 49 4b 27 7e 41 0a 31 a0 50 18	.>...XIK '~A.1.P.					
0030	07 d0 be 57 00 00 70 00 25 00 4e 5f a6 ca 00 00	...W..p. %N.....					
0040	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00					
0050	00 00 00 00 02 00 a1 00 04 00 01 00 fe 00 b1 00					
0060	11 00 04 00 cb 00 00 00 07 4d 00 f3 0a 60 05 4fM...'.0					
0070	10 02 00	...					

Figure 132. Protected Typed File Read Response over TCP (File Type Fuzzed)

C. PCCC PROTECTED TYPED FILE WRITE TEST CASES

This section shows the results of the PCCC Protected Typed File Write test cases.

(1) T56 Results

No.	Time	Source	Destination	Protocol	Length	Resp	Info
14	14:33:51.366668	192.168.0.59	192.168.0.62	CIP	115		Success: Class (0x67) - Service (0x4b)
15	14:33:51.376670	192.168.0.62	192.168.0.59	CIP	126		Class (0x67) - Service (0x4b)
16	14:33:51.386535	192.168.0.59	192.168.0.62	CIP	115		Success: Class (0x67) - Service (0x4b)
17	14:33:51.391172	192.168.0.62	192.168.0.59	CIP	126		Class (0x67) - Service (0x4b)
18	14:33:51.396611	192.168.0.59	192.168.0.62	CIP	115		Success: Class (0x67) - Service (0x4b)
19	14:33:51.404436	192.168.0.62	192.168.0.59	CIP	126		Class (0x67) - Service (0x4b)
▶ Frame 16: 115 bytes on wire (920 bits), 115 bytes captured (920 bits) on interface 0 ▶ Ethernet II, Src: Rockwell_a1:28:4c (00:1d:9c:a1:28:4c), Dst: Vmware_56:20:17 (00:0c:29:56:20:17) ▶ Internet Protocol Version 4, Src: 192.168.0.59, Dst: 192.168.0.62 ▶ Transmission Control Protocol, Src Port: 44818, Dst Port: 36560, Seq: 221, Ack: 333, Len: 61 ▶ EtherNet/IP (Industrial Protocol), Session: 0xBEEFF65, Send Unit Data ▶ Common Industrial Protocol ▶ CIP Class Generic							
0000	00 0c 29 56 20 17 00 1d 9c a1 28 4c 00 00 45 00	..)V(L..E.					
0010	00 65 38 e9 00 00 00 06 7f e0 c0 a0 00 3b c0 a8	.e8.....;..					
0020	00 3e af 12 0e d0 4d 54 9c 2e 50 b3 3d ad 50 18	.>....MT ..X.=.P.					
0030	07 d0 1f 2c 00 00 70 00 25 00 65 ff ef 8e 00 00p. %e.....					
0040	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00					
0050	00 00 00 00 02 00 a1 00 04 00 01 00 fe 00 b1 00					
0060	11 00 03 00 cb 00 00 00 07 4d 00 f3 0a 60 05 4fM...'.0					
0070	10 02 00	...					

Figure 133. Protected Typed File Write Response over TCP (Size Fuzzed)

(2) T57 Results

No.	Time	Source	Destination	Protocol	Length	Resp	Info
18	14:36:30.403153	192.168.0.59	192.168.0.62	CIP	115		Success: Class (0x67) - Service (0x4b)
19	14:36:30.413259	192.168.0.62	192.168.0.59	CIP	126		Class (0x67) - Service (0x4b)
20	14:36:30.424488	192.168.0.59	192.168.0.62	CIP	115		Success: Class (0x67) - Service (0x4b)
21	14:36:30.430687	192.168.0.62	192.168.0.59	CIP	126		Class (0x67) - Service (0x4b)
22	14:36:30.443140	192.168.0.59	192.168.0.62	CIP	115		Success: Class (0x67) - Service (0x4b)
23	14:36:30.452073	192.168.0.62	192.168.0.59	CIP	126		Class (0x67) - Service (0x4b)
▶ Frame 20: 115 bytes on wire (920 bits), 115 bytes captured (920 bits) on interface 0 ▶ Ethernet II, Src: Rockwell_al:28:4c (00:1d:9c:a1:28:4c), Dst: Vmware_56:20:17 (00:0c:29:56:20:17) ▶ Internet Protocol Version 4, Src: 192.168.0.59, Dst: 192.168.0.62 ▶ Transmission Control Protocol, Src Port: 44818, Dst Port: 36562, Seq: 404, Ack: 549, Len: 61 ▶ Ethernet/IP (Industrial Protocol), Session: 0x08CB3684, Send Unit Data ▶ Common Industrial Protocol ▼ CIP Class Generic ▼ Command Specific Data Data: 074d00f30a60054f100200							
0000	00 0c 29 56 20 17 00 1d 9c a1 28 4c 00 00 45 00	..)V(L..E.					
0010	00 65 41 b3 00 00 00 06 77 16 c0 a8 00 3b c0 a8	.eA.....W....;..					
0020	00 3e af 12 8e d2 49 1b 35 43 a6 ea 40 e2 50 18	.>...I. 5C...@.P.					
0030	07 d0 0b 61 00 00 70 00 25 00 b4 36 cb 08 00 00	...a..p. %.6....					
0040	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00					
0050	00 00 00 00 02 00 a1 00 04 00 01 00 fe 80 b1 00					
0060	11 00 06 00 cb 00 00 00 07 4d 00 f3 0a 60 05 4fM...`0					
0070	10 02 00	...					

Figure 134. Protected Typed File Write Response over TCP (Tag Fuzzed)

(3) T58 Results

No.	Time	Source	Destination	Protocol	Length	Resp	Info
11	14:40:43.683827	192.168.0.62	192.168.0.59	CIP	126		Class (0x67) - Service (0x4b)
12	14:40:43.687431	192.168.0.59	192.168.0.62	CIP	115		Success: Class (0x67) - Service (0x4b)
13	14:40:43.697450	192.168.0.62	192.168.0.59	CIP	126		Class (0x67) - Service (0x4b)
14	14:40:43.707535	192.168.0.59	192.168.0.62	CIP	115		Success: Class (0x67) - Service (0x4b)
15	14:40:43.717500	192.168.0.62	192.168.0.59	CIP	126		Class (0x67) - Service (0x4b)
16	14:40:43.727834	192.168.0.59	192.168.0.62	CIP	115		Success: Class (0x67) - Service (0x4b)
▶ Frame 14: 115 bytes on wire (920 bits), 115 bytes captured (920 bits) on interface 0 ▶ Ethernet II, Src: Rockwell_al:28:4c (00:1d:9c:a1:28:4c), Dst: Vmware_56:20:17 (00:0c:29:56:20:17) ▶ Internet Protocol Version 4, Src: 192.168.0.59, Dst: 192.168.0.62 ▶ Transmission Control Protocol, Src Port: 44818, Dst Port: 36564, Seq: 221, Ack: 333, Len: 61 ▶ Ethernet/IP (Industrial Protocol), Session: 0x0191FAB6, Send Unit Data ▶ Common Industrial Protocol ▼ CIP Class Generic ▼ Command Specific Data Data: 074d00f30a60054f100200							
0000	00 0c 29 56 20 17 00 1d 9c a1 28 4c 00 00 45 00	..)V(L..E.					
0010	00 65 64 c3 00 00 00 06 54 06 c0 a8 00 3b c0 a8	.eD.....T....;..					
0020	00 3e af 12 8e d4 4d 57 e7 c1 a9 ad 50 e8 50 18	.>...MWX.P.					
0030	07 d0 74 ee 00 00 70 00 25 00 b6 fa 91 01 00 00	..t...p. %......					
0040	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00					
0050	00 00 00 00 02 00 a1 00 04 00 01 00 fe 80 b1 00					
0060	11 00 03 00 cb 00 00 00 07 4d 00 f3 0a 60 05 4fM...`0					
0070	10 02 00	...					

Figure 135. Protected Typed File Write Response over TCP (Offset Fuzzed)

(4) T59 Results

No.	Time	Source	Destination	Protocol	Length	Resp	Info
13	14:46:12.129941	192.168.0.62	192.168.0.59	CIP	126		Class (0x67) - Service (0x4b)
14	14:46:12.140355	192.168.0.59	192.168.0.62	CIP	115		Success: Class (0x67) - Service (0x4b)
15	14:46:12.145089	192.168.0.62	192.168.0.59	CIP	126		Class (0x67) - Service (0x4b)
16	14:46:12.150543	192.168.0.59	192.168.0.62	CIP	115		Success: Class (0x67) - Service (0x4b)
17	14:46:12.160070	192.168.0.62	192.168.0.59	CIP	126		Class (0x67) - Service (0x4b)
18	14:46:12.170478	192.168.0.59	192.168.0.62	CIP	115		Success: Class (0x67) - Service (0x4b)
▶ Frame 16: 115 bytes on wire (920 bits), 115 bytes captured (920 bits) on interface 0 ▶ Ethernet II, Src: Rockwell_a1:28:4c (00:1d:9c:a1:28:4c), Dst: Vmware_56:20:17 (00:0c:29:56:20:17) ▶ Internet Protocol Version 4, Src: 192.168.0.59, Dst: 192.168.0.62 ▶ Transmission Control Protocol, Src Port: 44818, Dst Port: 36566, Seq: 282, Ack: 405, Len: 61 ▶ Ethernet/IP (Industrial Protocol), Session: 0xB422F6CE, Send Unit Data ▶ Common Industrial Protocol ▼ CIP Class Generic ▼ Command Specific Data Data: 074d00f30a60054f100200							
0000	00 0c 29 56 20 17 00 1d 9c a1 28 4c 08 00 45 00	..)V(L..E.					
0010	00 65 82 a8 00 00 00 06 36 21 c0 a8 00 3b c0 a8	.e..... 6!...;..					
0020	00 3e af 12 8e d6 49 24 a4 2b c1 a8 b2 79 50 18	.>....15 .+...yP.					
0030	07 d0 a0 7a 00 00 70 00 25 00 ce f6 22 b4 00 00	...z..p. %..P..					
0040	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00					
0050	00 00 00 00 02 00 a1 00 04 00 01 00 fe 08 b1 00					
0060	11 00 04 00 cb 00 00 00 07 4d 00 f3 0a 60 05 4fM...'.0					
0070	10 02 00	...					

Figure 136. Protected Typed File Write Response over TCP (File Type Fuzzed)

(5) T60 Results

No.	Time	Source	Destination	Protocol	Length	Resp	Info
19	14:23:22.938944	192.168.0.62	192.168.0.59	CIP	126		Class (0x67) - Service (0x4b)
20	14:23:22.949794	192.168.0.59	192.168.0.62	CIP	115		Success: Class (0x67) - Service (0x4b)
21	14:23:22.962069	192.168.0.62	192.168.0.59	CIP	126		Class (0x67) - Service (0x4b)
22	14:23:22.969514	192.168.0.59	192.168.0.62	CIP	115		Success: Class (0x67) - Service (0x4b)
23	14:23:22.974942	192.168.0.62	192.168.0.59	CIP	126		Class (0x67) - Service (0x4b)
24	14:23:22.979326	192.168.0.59	192.168.0.62	CIP	115		Success: Class (0x67) - Service (0x4b)
▶ Frame 20: 115 bytes on wire (920 bits), 115 bytes captured (920 bits) on interface 0 ▶ Ethernet II, Src: Rockwell_a1:28:4c (00:1d:9c:a1:28:4c), Dst: Vmware_56:20:17 (00:0c:29:56:20:17) ▶ Internet Protocol Version 4, Src: 192.168.0.59, Dst: 192.168.0.62 ▶ Transmission Control Protocol, Src Port: 44818, Dst Port: 36558, Seq: 282, Ack: 405, Len: 61 ▶ Ethernet/IP (Industrial Protocol), Session: 0x45300586, Send Unit Data ▶ Common Industrial Protocol ▼ CIP Class Generic ▼ Command Specific Data Data: 074d00f30a60054f100200							
0000	00 0c 29 56 20 17 00 1d 9c a1 28 4c 08 00 45 00	..)V(L..E.					
0010	00 65 00 57 00 00 00 06 b0 72 c0 a8 00 3b c0 a8	.e.W.... .r...;..					
0020	00 3e af 12 8e ce 49 0c 81 50 78 b9 5f 59 50 18	.>....I. .Px..yP.					
0030	07 d0 9a e5 00 00 70 00 25 00 06 05 30 45 00 00p. %...0E..					
0040	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00					
0050	00 00 00 00 02 00 a1 00 04 00 01 00 fe 08 b1 00					
0060	11 00 04 00 cb 00 00 00 07 4d 00 f3 0a 60 05 4fM...'.0					
0070	10 02 00	...					

Figure 137. Protected Typed File Write Response over TCP (Data Fuzzed)

D. PCCC PROTECTED LOGICAL WRITE WITH THREE ADDRESS FIELDS TEST CASES

This section shows the results of the PCCC Protected Logical Write with Three Address Fields test cases.

(1) T61 Results

The Protected Logical Write with Three Address Fields responds with an EXT STS of 0x0B (“access denied, improper privilege”) when Byte Size is set to 0x00 (Figure

138). All other Byte Size inputs return responses with STS of 0x10 (“illegal command or format”) as demonstrated in Figure 139.

No.	Time	Source	Destination	Protocol	Length	Resp	Info
1172	20:18:22.017135	192.168.0.59	192.168.0.62	CIP	115		Success: Class (0x67) - Service (0x4b)
1173	20:18:22.022435	192.168.0.62	192.168.0.59	CIP	123		Class (0x67) - Service (0x4b)
1174	20:18:22.036986	192.168.0.59	192.168.0.62	CIP	116		Success: Class (0x67) - Service (0x4b)
1175	20:18:22.048965	192.168.0.62	192.168.0.59	CIP	123		Class (0x67) - Service (0x4b)
1176	20:18:22.056812	192.168.0.59	192.168.0.62	CIP	115		Success: Class (0x67) - Service (0x4b)
1177	20:18:22.062248	192.168.0.62	192.168.0.59	CIP	123		Class (0x67) - Service (0x4b)
▶ Frame 1174: 116 bytes on wire (928 bits), 116 bytes captured (928 bits) on interface 0 ▶ Ethernet II, Src: Rockwell_a1:28:4c (00:1d:9c:a1:28:4c), Dst: Vmware_56:20:17 (00:0c:29:56:20:17) ▶ Internet Protocol Version 4, Src: 192.168.0.59, Dst: 192.168.0.62 ▶ Transmission Control Protocol, Src Port: 44818, Dst Port: 41994, Seq: 35541, Ack: 40273, Len: 62 ▶ EtherNet/IP (Industrial Protocol), Session: 0x30821425, Send Unit Data ▶ Common Industrial Protocol ▼ CIP Class Generic ▼ Command Specific Data Data: 074d00f30a60054ff047020b							
0000	00 0c 29 56 20 17 00 1d 9c a1 28 4c 00 00 45 00	..)V(L..E.					
0010	00 66 33 a6 00 00 00 06 85 22 c0 a8 00 3b c0 a8	.f3....".....					
0020	00 3e af 12 a4 0a 4d 66 14 25 17 bc e7 3f 50 18	.>...Mf %....7P.					
0030	07 d0 00 43 00 00 70 00 26 00 25 14 b2 30 00 00	...C.p. &%.0..					
0040	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00					
0050	00 00 00 00 02 00 a1 00 04 00 01 00 fe 00 b1 00					
0060	12 00 46 02 cb 00 00 00 07 4d 00 f3 0a 60 05 4f	..F.....M...".0					
0070	f0 47 02 0b	.G..					

Figure 138. Protected Logical Write with Three Address Fields Response over TCP (Byte Size 0x00)

No.	Time	Source	Destination	Protocol	Length	Resp	Info
11	20:18:10.721933	192.168.0.62	192.168.0.59	CIP	123		Class (0x67) - Service (0x4b)
12	20:18:10.726771	192.168.0.59	192.168.0.62	CIP	115		Success: Class (0x67) - Service (0x4b)
13	20:18:10.738763	192.168.0.62	192.168.0.59	CIP	123		Class (0x67) - Service (0x4b)
14	20:18:10.746927	192.168.0.59	192.168.0.62	CIP	115		Success: Class (0x67) - Service (0x4b)
▶ Frame 12: 115 bytes on wire (920 bits), 115 bytes captured (920 bits) on interface 0 ▶ Ethernet II, Src: Rockwell_a1:28:4c (00:1d:9c:a1:28:4c), Dst: Vmware_56:20:17 (00:0c:29:56:20:17) ▶ Internet Protocol Version 4, Src: 192.168.0.59, Dst: 192.168.0.62 ▶ Transmission Control Protocol, Src Port: 44818, Dst Port: 41994, Seq: 160, Ack: 253, Len: 51 ▶ EtherNet/IP (Industrial Protocol), Session: 0x30821425, Send Unit Data ▶ Common Industrial Protocol ▼ CIP Class Generic ▼ Command Specific Data Data: 074d00f30a60054f100300							
0000	00 0c 29 56 20 17 00 1d 9c a1 28 4c 00 00 45 00	..)V(L..E.					
0010	00 65 31 62 00 00 00 06 87 67 c0 a8 00 3b c0 a8	.e1b....g....					
0020	00 3e af 12 a4 0a 4d 65 89 f0 17 bc 4a eb 50 18	.>...MeJ.P.					
0030	07 d0 cf 20 00 00 70 00 25 00 25 14 b2 30 00 00p. &%.0..					
0040	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00					
0050	00 00 00 00 02 00 a1 00 04 00 01 00 fe 00 b1 00					
0060	11 00 02 00 cb 00 00 00 07 4d 00 f3 0a 60 05 4fM...".0					
0070	10 03 00	...					

Figure 139. Protected Logical Write with Three Address Fields Response over TCP (Byte Size Fuzzed)

(2) T62 Results

No.	Time	Source	Destination	Protocol	Length	Resp	Info
20	20:29:42.793660	192.168.0.59	192.168.0.62	CIP	115		Success: Class (0x67) - Service (0x4b)
21	20:29:42.801690	192.168.0.62	192.168.0.59	CIP	123		Class (0x67) - Service (0x4b)
22	20:29:42.813527	192.168.0.59	192.168.0.62	CIP	115		Success: Class (0x67) - Service (0x4b)
23	20:29:42.827561	192.168.0.62	192.168.0.59	CIP	123		Class (0x67) - Service (0x4b)
▶ Frame 22: 115 bytes on wire (920 bits), 115 bytes captured (920 bits) on interface 0 ▶ Ethernet II, Src: Rockwell_a1:28:4c (00:1d:9c:a1:28:4c), Dst: Vmware_56:20:17 (00:0c:29:56:20:17) ▶ Internet Protocol Version 4, Src: 192.168.0.59, Dst: 192.168.0.62 ▶ Transmission Control Protocol, Src Port: 44818, Dst Port: 42002, Seq: 465, Ack: 598, Len: 61 ▶ EtherNet/IP (Industrial Protocol), Session: 0x444E6405, Send Unit Data ▶ Common Industrial Protocol ▼ CIP Class Generic ▼ Command Specific Data Data: 074d00f30a60054f100000							
0000	00 0c 29 56 20 17 00 1d	9c a1 28 4c 08 00 45 00	..)V(L..E.				
0010	00 65 37 b5 00 00 00 06	01 14 c0 a8 00 3b c0 a8	.e7.....;..				
0020	00 3e af 12 a4 12 4d 68	bc ec f8 0b 04 a5 50 18	.>....MhP.				
0030	07 d0 00 a7 00 00 70 00	25 00 05 64 4e 44 00 00p. %..dND..				
0040	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00				
0050	00 00 00 00 02 00 a1 00	04 00 01 00 fe 00 b1 00				
0060	11 00 07 00 cb 00 00 00	07 4d 00 f3 0a 60 05 4fM....'0				
0070	10 08 00		...				

Figure 140. Protected Logical Write with Three Address Fields Response over TCP (File No. Fuzzed)

(3) T63 Results

No.	Time	Source	Destination	Protocol	Length	Resp	Info
20	20:29:42.793660	192.168.0.59	192.168.0.62	CIP	115		Success: Class (0x67) - Service (0x4b)
21	20:29:42.801690	192.168.0.62	192.168.0.59	CIP	123		Class (0x67) - Service (0x4b)
22	20:29:42.813527	192.168.0.59	192.168.0.62	CIP	115		Success: Class (0x67) - Service (0x4b)
23	20:29:42.827561	192.168.0.62	192.168.0.59	CIP	123		Class (0x67) - Service (0x4b)
▶ Frame 22: 115 bytes on wire (920 bits), 115 bytes captured (920 bits) on interface 0 ▶ Ethernet II, Src: Rockwell_a1:28:4c (00:1d:9c:a1:28:4c), Dst: Vmware_56:20:17 (00:0c:29:56:20:17) ▶ Internet Protocol Version 4, Src: 192.168.0.59, Dst: 192.168.0.62 ▶ Transmission Control Protocol, Src Port: 44818, Dst Port: 42002, Seq: 465, Ack: 598, Len: 61 ▶ EtherNet/IP (Industrial Protocol), Session: 0x444E6405, Send Unit Data ▶ Common Industrial Protocol ▼ CIP Class Generic ▼ Command Specific Data Data: 074d00f30a60054f100000							
0000	00 0c 29 56 20 17 00 1d	9c a1 28 4c 08 00 45 00	..)V(L..E.				
0010	00 65 37 b5 00 00 00 06	01 14 c0 a8 00 3b c0 a8	.e7.....;..				
0020	00 3e af 12 a4 12 4d 68	bc ec f8 0b 04 a5 50 18	.>....MhP.				
0030	07 d0 00 a7 00 00 70 00	25 00 05 64 4e 44 00 00p. %..dND..				
0040	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00				
0050	00 00 00 00 02 00 a1 00	04 00 01 00 fe 00 b1 00				
0060	11 00 07 00 cb 00 00 00	07 4d 00 f3 0a 60 05 4fM....'0				
0070	10 08 00		...				

Figure 141. Protected Logical Write with Three Address Fields Response over TCP (File Type Fuzzed)

(4) T64 Results

No.	Time	Source	Destination	Protocol	Length	Resp	Info
12	20:31:16.401729	192.168.0.59	192.168.0.62	CIP	115		Success: Class (0x67) - Service (0x4b)
13	20:31:16.412216	192.168.0.62	192.168.0.59	CIP	123		Class (0x67) - Service (0x4b)
14	20:31:16.421989	192.168.0.59	192.168.0.62	CIP	115		Success: Class (0x67) - Service (0x4b)
15	20:31:16.426812	192.168.0.62	192.168.0.59	CIP	123		Class (0x67) - Service (0x4b)
▶ Frame 14: 115 bytes on wire (920 bits), 115 bytes captured (920 bits) on interface 0 ▶ Ethernet II, Src: Rockwell_a1:28:4c (00:1d:9c:a1:28:4c), Dst: Vmware_56:20:17 (00:0c:29:56:20:17) ▶ Internet Protocol Version 4, Src: 192.168.0.59, Dst: 192.168.0.62 ▶ Transmission Control Protocol, Src Port: 44818, Dst Port: 42004, Seq: 221, Ack: 322, Len: 61 ▶ Ethernet/IP (Industrial Protocol), Session: 0x004C1A47, Send Unit Data ▶ Common Industrial Protocol ▼ CIP Class Generic ▼ Command Specific Data Data: 074d00f30a60054f100400							
0000	00 0c 29 56 20 17 00 1d	9c a1 28 4c 00 00 45 00	..)V(L..E.				
0010	00 65 3a 40 00 00 00 06	7e 89 c0 a0 00 3b c0 a0	.e@....~....;..				
0020	00 3e af 12 a4 14 49 1e	ed 78 39 c0 ce 7f 50 18	.>...I. .x9...P.				
0030	07 d0 0d 66 00 00 70 00	25 00 47 1a 4c 00 00 00	...f..p. %G.L...				
0040	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00				
0050	00 00 00 00 02 00 a1 00	04 00 01 00 fe 80 b1 00				
0060	11 00 03 00 cb 00 00 00	07 4d 00 f3 0a 60 05 4fM...'.0				
0070	10 04 00		...				

Figure 142. Protected Logical Write with Three Address Fields Response over TCP (Element No. Fuzzed)

(5) T65 Results

No.	Time	Source	Destination	Protocol	Length	Resp	Info
16	20:32:31.620250	192.168.0.59	192.168.0.62	CIP	115		Success: Class (0x67) - Service (0x4b)
17	20:32:31.627311	192.168.0.62	192.168.0.59	CIP	123		Class (0x67) - Service (0x4b)
18	20:32:31.640424	192.168.0.59	192.168.0.62	CIP	115		Success: Class (0x67) - Service (0x4b)
19	20:32:31.651088	192.168.0.62	192.168.0.59	CIP	123		Class (0x67) - Service (0x4b)
20	20:32:31.660451	192.168.0.59	192.168.0.62	CIP	115		Success: Class (0x67) - Service (0x4b)
21	20:32:31.669719	192.168.0.62	192.168.0.59	CIP	123		Class (0x67) - Service (0x4b)
▶ Frame 18: 115 bytes on wire (920 bits), 115 bytes captured (920 bits) on interface 0 ▶ Ethernet II, Src: Rockwell_a1:28:4c (00:1d:9c:a1:28:4c), Dst: Vmware_56:20:17 (00:0c:29:56:20:17) ▶ Internet Protocol Version 4, Src: 192.168.0.59, Dst: 192.168.0.62 ▶ Transmission Control Protocol, Src Port: 44818, Dst Port: 42006, Seq: 343, Ack: 460, Len: 61 ▶ Ethernet/IP (Industrial Protocol), Session: 0x6F90EB53, Send Unit Data ▶ Common Industrial Protocol ▼ CIP Class Generic ▼ Command Specific Data Data: 074d00f30a60054f100600							
0000	00 0c 29 56 20 17 00 1d	9c a1 28 4c 00 00 45 00	..)V(L..E.				
0010	00 65 3d 51 00 00 00 06	7b 78 c0 a0 00 3b c0 a0	.e@Q...{x...;..				
0020	00 3e af 12 a4 16 4d 6a	bc 4b 46 91 e0 f4 50 18	.>...M}..HF...P.				
0030	07 d0 bf bf 00 00 70 00	25 00 53 eb 90 6f 00 00	...p. %S..O...				
0040	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00				
0050	00 00 00 00 02 00 a1 00	04 00 01 00 fe 80 b1 00				
0060	11 00 05 00 cb 00 00 00	07 4d 00 f3 0a 60 05 4fM...'.0				
0070	10 06 00		...				

Figure 143. Protected Logical Write with Three Address Fields Response over TCP (Sub-Element No. Fuzzed)

E. PCCC UNPROTECTED READ TEST CASES

This section shows the results of the PCCC Unprotected Read test cases.

(1) T66 Results

No.	Time	Source	Destination	Protocol	Length	Resp	Info
18	15:30:26.024416	192.168.0.59	192.168.0.62	CIP	115		Success: Class (0x67) - Service (0x4b)
19	15:30:26.029348	192.168.0.62	192.168.0.59	CIP	120		Class (0x67) - Service (0x4b)
20	15:30:26.034062	192.168.0.59	192.168.0.62	CIP	115		Success: Class (0x67) - Service (0x4b)
21	15:30:26.043746	192.168.0.62	192.168.0.59	CIP	120		Class (0x67) - Service (0x4b)
22	15:30:26.053022	192.168.0.59	192.168.0.62	CIP	115		Success: Class (0x67) - Service (0x4b)
23	15:30:26.061894	192.168.0.62	192.168.0.59	CIP	120		Class (0x67) - Service (0x4b)
▶ Frame 20: 115 bytes on wire (920 bits), 115 bytes captured (920 bits) on interface 0							
▶ Ethernet II, Src: Rockwell_a1:28:4c (00:1d:9c:a1:28:4c), Dst: Vmware_56:20:17 (00:0c:29:56:20:17)							
▶ Internet Protocol Version 4, Src: 192.168.0.59, Dst: 192.168.0.62							
▶ Transmission Control Protocol, Src Port: 44818, Dst Port: 60846, Seq: 282, Ack: 381, Len: 61							
▶ EtherNet/IP (Industrial Protocol), Session: 0xE78D8FB5, Send Unit Data							
▶ Common Industrial Protocol							
▼ CIP Class Generic							
▼ Command Specific Data							
Data: 074d00f30a600541100200							
0000	00 0c 29 56 20 17 00 1d 9c a1 28 4c 00 00 45 00	..)V(L..E.					
0010	00 65 05 de 00 00 00 06 b2 eb c0 a8 00 3b c0 a8	.e.....					
0020	00 3e af 12 ed ae 49 10 3a e7 a8 40 5d c8 50 18	.>....I. :..@].P.					
0030	07 d0 c7 55 00 00 70 00 25 00 b5 8f 8d e7 00 00	...U..p. %.....					
0040	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00					
0050	00 00 00 00 02 00 a1 00 04 00 01 00 fe 00 b1 00					
0060	11 00 04 00 cb 00 00 00 07 4d 00 f3 0a 60 05 41M...`A					
0070	10 02 00	...					

Figure 144. Unprotected Read Response over TCP (Address Fuzzed)

(2) T67 Results

No.	Time	Source	Destination	Protocol	Length	Resp	Info
19	15:40:46.171926	192.168.0.62	192.168.0.59	CIP	120		Class (0x67) - Service (0x4b)
20	15:40:46.182486	192.168.0.59	192.168.0.62	CIP	115		Success: Class (0x67) - Service (0x4b)
21	15:40:46.188186	192.168.0.62	192.168.0.59	CIP	120		Class (0x67) - Service (0x4b)
22	15:40:46.193216	192.168.0.59	192.168.0.62	CIP	115		Success: Class (0x67) - Service (0x4b)
23	15:40:46.211240	192.168.0.62	192.168.0.59	CIP	120		Class (0x67) - Service (0x4b)
24	15:40:46.222319	192.168.0.59	192.168.0.62	CIP	115		Success: Class (0x67) - Service (0x4b)
▶ Frame 22: 115 bytes on wire (920 bits), 115 bytes captured (920 bits) on interface 0							
▶ Ethernet II, Src: Rockwell_a1:28:4c (00:1d:9c:a1:28:4c), Dst: Vmware_56:20:17 (00:0c:29:56:20:17)							
▶ Internet Protocol Version 4, Src: 192.168.0.59, Dst: 192.168.0.62							
▶ Transmission Control Protocol, Src Port: 44818, Dst Port: 60848, Seq: 465, Ack: 579, Len: 61							
▶ EtherNet/IP (Industrial Protocol), Session: 0xB189D787, Send Unit Data							
▶ Common Industrial Protocol							
▼ CIP Class Generic							
▼ Command Specific Data							
Data: 074d00f30a600541100200							
0000	00 0c 29 56 20 17 00 1d 9c a1 28 4c 00 00 45 00	..)V(L..E.					
0010	00 65 78 ba 00 00 00 06 40 0f c0 a8 00 3b c0 a8	.ex..... @.....					
0020	00 3e af 12 ed b0 4d 58 dd 10 7a 07 40 36 50 18	.>....MX :z.@6P.					
0030	07 d0 9b 1b 00 00 70 00 25 00 87 d7 99 b1 00 00P. %.....					
0040	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00					
0050	00 00 00 00 02 00 a1 00 04 00 01 00 fe 00 b1 00					
0060	11 00 07 00 cb 00 00 00 07 4d 00 f3 0a 60 05 41M...`A					
0070	10 02 00	...					

Figure 145. Unprotected Read Response over TCP (Size Fuzzed).

F. PCCC DIAGNOSTIC STATUS TEST CASES

This section shows the results of the PCCC Diagnostic Status test cases.

(1) T68 Results

No.	Time	Source	Destination	Protocol	Length	Resp	Info
30	13:49:52.586635	192.168.0.59	192.168.0.62	CIP	148		Success: Class (0x67) - Service (0x4b)
31	13:49:52.592078	192.168.0.62	192.168.0.59	CIP	118		Class (0x67) - Service (0x4b)
32	13:49:52.596225	192.168.0.59	192.168.0.62	CIP	148		Success: Class (0x67) - Service (0x4b)
33	13:49:52.605056	192.168.0.62	192.168.0.59	CIP	118		Class (0x67) - Service (0x4b)
▶ Frame 32: 148 bytes on wire (1120 bits), 148 bytes captured (1120 bits) on interface 0 ▶ Ethernet II, Src: Rockwell_a1:28:4c (00:1d:9c:a1:28:4c), Dst: Vmware_56:20:17 (00:0c:29:56:20:17) ▶ Internet Protocol Version 4, Src: 192.168.0.59, Dst: 192.168.0.62 ▶ Transmission Control Protocol, Src Port: 44818, Dst Port: 54856, Seq: 443, Ack: 435, Len: 86 ▶ Ethernet/IP (Industrial Protocol), Session: 0xA4E4A1DF, Send Unit Data ▶ Common Industrial Protocol ▼ CIP Class Generic ▼ Command Specific Data Data: 074d00f30a60054600020000ee4a9c23313736332d4c4543...							
0000	00 0c 29 56 20 17 00 1d 9c a1 28 4c 00 00 45 00	..)V(L..E.					
0010	00 7e 64 0e 00 00 00 06 54 22 c0 a0 00 3b c0 a8	.~d.... T^....;..					
0020	00 3e af 12 d6 48 49 24 dd 48 d2 4d f2 d4 50 18	.>...HI\$.H.M..P.					
0030	07 d0 22 a6 00 00 70 00 3e 00 df a1 e4 aa 00 00	.."...p. >.....					
0040	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00					
0050	00 00 00 00 02 00 a1 00 04 00 01 00 fe 00 b1 00					
0060	2a 00 05 00 cb 00 00 00 07 4d 00 f3 0a 60 05 46	*..... .M....F					
0070	00 02 00 00 ee 4a 9c 23 31 37 36 33 2d 4c 45 43J.# 1763-LEC					
0080	20 20 20 00 00 3e 00 9a 15 30 fc 01	..>... .0..					

Figure 146. Diagnostic Status Response over TCP (Functionality Test)

G. PCCC READ DIAGNOSTIC COUNTERS TEST CASES

This section shows the results of the PCCC Read Diagnostic Counters test cases.

(1) T69 Results

No.	Time	Source	Destination	Protocol	Length	Resp	Info
22	10:30:26.072681	192.168.0.59	192.168.0.62	CIP	115		Success: Class (0x67) - Service (0x4b)
23	10:30:26.082111	192.168.0.62	192.168.0.59	CIP	121		Class (0x67) - Service (0x4b)
24	10:30:26.093041	192.168.0.59	192.168.0.62	CIP	115		Success: Class (0x67) - Service (0x4b)
25	10:30:26.104645	192.168.0.62	192.168.0.59	CIP	121		Class (0x67) - Service (0x4b)
▶ Frame 24: 115 bytes on wire (920 bits), 115 bytes captured (920 bits) on interface 0 ▶ Ethernet II, Src: Rockwell_a1:28:4c (00:1d:9c:a1:28:4c), Dst: Vmware_56:20:17 (00:0c:29:56:20:17) ▶ Internet Protocol Version 4, Src: 192.168.0.59, Dst: 192.168.0.62 ▶ Transmission Control Protocol, Src Port: 44818, Dst Port: 60206, Seq: 526, Ack: 653, Len: 61 ▶ Ethernet/IP (Industrial Protocol), Session: 0xf17FC017, Send Unit Data ▶ Common Industrial Protocol ▼ CIP Class Generic ▼ Command Specific Data Data: 074d00f30a600546100200							
0000	00 0c 29 56 20 17 00 1d 9c a1 28 4c 00 00 45 00	..)V(L..E.					
0010	00 65 00 ab 00 00 00 06 b8 1e c0 a0 00 3b c0 a8	.e..... :..					
0020	00 3e af 12 eb 2e 49 0c 97 f3 0a 73 a9 0c 50 18	.>....I. ...S.P.					
0030	07 d0 67 17 00 00 70 00 25 00 17 c0 7f f1 00 00	..g...p. %.....					
0040	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00					
0050	00 00 00 00 02 00 a1 00 04 00 01 00 fe 00 b1 00					
0060	11 00 00 00 cb 00 00 00 07 4d 00 f3 0a 60 05 46M....F					
0070	10 02 00	...					

Figure 147. Read Diagnostic Counters Response over TCP
(Address Fuzzed: 0x3455)

(2) T70 Results

No.	Time	Source	Destination	Protocol	Length	Resp	Info
17	10:52:51.534022	192.168.0.62	192.168.0.59	CIP	121		Class (0x67) - Service (0x4b)
18	10:52:51.546896	192.168.0.59	192.168.0.62	CIP	152		Success: Class (0x67) - Service (0x4b)
19	10:52:51.553420	192.168.0.62	192.168.0.59	CIP	121		Class (0x67) - Service (0x4b)
20	10:52:51.556888	192.168.0.59	192.168.0.62	CIP	115		Success: Class (0x67) - Service (0x4b)
▶ Frame 18: 152 bytes on wire (1216 bits), 152 bytes captured (1216 bits) on interface 0 ▶ Ethernet II, Src: Rockwell_a1:28:4c (00:1d:9c:a1:28:4c), Dst: Vmware_56:20:17 (00:0c:29:56:20:17) ▶ Internet Protocol Version 4, Src: 192.168.0.59, Dst: 192.168.0.62 ▶ Transmission Control Protocol, Src Port: 44818, Dst Port: 60208, Seq: 515, Ack: 452, Len: 98 ▶ Ethernet/IP (Industrial Protocol), Session: 0xb1EE95FC, Send Unit Data ▶ Common Industrial Protocol ▼ CIP Class Generic ▼ Command Specific Data Data: 074d00f30a60054600020055ea7d009d6377008302010005...							
0000	00 0c 29 56 20 17 00 1d	9c a1 28 4c 00 00 45 00	..)V(L..E.				
0010	00 8a 00 ad 00 00 00 06	b7 f7 c0 a8 00 3b c0 a8;..				
0020	00 3e af 12 eb 30 4d 54	99 fe ef 47 07 83 50 18	.>...0MT ...G..P.				
0030	07 d0 40 4a 00 00 70 00	4a 00 fc 95 ee b1 00 00	..@J..p. J.....				
0040	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00				
0050	00 00 00 00 02 00 a1 00	04 00 01 00 fe 00 b1 00				
0060	36 00 05 00 cb 00 00 00	07 4d 00 f3 0a 60 05 46	6.....M...F				
0070	00 02 00 55 ea 7d 00 9d	63 77 00 83 02 01 00 05	...U.J.. CW.....				
0080	01 01 00 00 00 00 00 00	00 00 00 00 00 00 00 00				
0090	00 00 00 00 00 00 00 00					

Figure 148. Read Diagnostic Counters Response over TCP (Size Fuzzed: 25)

No.	Time	Source	Destination	Protocol	Length	Resp	Info
17	10:52:51.534022	192.168.0.62	192.168.0.59	CIP	121		Class (0x67) - Service (0x4b)
18	10:52:51.546896	192.168.0.59	192.168.0.62	CIP	152		Success: Class (0x67) - Service (0x4b)
19	10:52:51.553420	192.168.0.62	192.168.0.59	CIP	121		Class (0x67) - Service (0x4b)
20	10:52:51.556888	192.168.0.59	192.168.0.62	CIP	115		Success: Class (0x67) - Service (0x4b)
21	10:52:51.567521	192.168.0.62	192.168.0.59	CIP	121		Class (0x67) - Service (0x4b)
▶ Frame 20: 115 bytes on wire (920 bits), 115 bytes captured (920 bits) on interface 0 ▶ Ethernet II, Src: Rockwell_a1:28:4c (00:1d:9c:a1:28:4c), Dst: Vmware_56:20:17 (00:0c:29:56:20:17) ▶ Internet Protocol Version 4, Src: 192.168.0.59, Dst: 192.168.0.62 ▶ Transmission Control Protocol, Src Port: 44818, Dst Port: 60208, Seq: 613, Ack: 519, Len: 61 ▶ Ethernet/IP (Industrial Protocol), Session: 0xb1EE95FC, Send Unit Data ▶ Common Industrial Protocol ▼ CIP Class Generic ▼ Command Specific Data Data: 074d00f30a600546100200							
0000	00 0c 29 56 20 17 00 1d	9c a1 28 4c 00 00 45 00	..)V(L..E.				
0010	00 65 00 ae 00 00 00 06	b8 1b c0 a8 00 3b c0 a8	e.....;..				
0020	00 3e af 12 eb 30 4d 54	9a 60 ef 47 07 c6 50 18	.>...0MT ...G..P.				
0030	07 d0 cb 3b 00 00 70 00	25 00 fc 95 ee b1 00 00	...;..p. %.....				
0040	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00				
0050	00 00 00 00 02 00 a1 00	04 00 01 00 fe 00 b1 00				
0060	11 00 06 00 cb 00 00 00	07 4d 00 f3 0a 60 05 46M...F				
0070	10 02 00		...				

Figure 149. Read Diagnostic Counters Response over TCP (Size Fuzzed: 75)

H. PCCC RESTART TEST CASES

This section shows the results of the PCCC Restart test cases.

(1) T71 Results

No.	Time	Source	Destination	Protocol	Length	Resp	Info
14	15:36:22.245719	192.168.0.59	192.168.0.62	CIP CM	124		Success: Connection Manager - Forward Open
15	15:36:22.257846	192.168.0.62	192.168.0.59	CIP	118		Class (0x67) - Service (0x4b)
16	15:36:22.266453	192.168.0.59	192.168.0.62	CIP	115		Success: Class (0x67) - Service (0x4b)
17	15:36:22.270764	192.168.0.62	192.168.0.59	CIP	118		Class (0x67) - Service (0x4b)
▶ Frame 16: 115 bytes on wire (920 bits), 115 bytes captured (920 bits) on interface 0							
▶ Ethernet II, Src: Rockwell_a1:28:4c (00:1d:9c:a1:28:4c), Dst: Vmware_56:20:17 (00:0c:29:56:20:17)							
▶ Internet Protocol Version 4, Src: 192.168.0.59, Dst: 192.168.0.62							
▶ Transmission Control Protocol, Src Port: 44818, Dst Port: 54880, Seq: 99, Ack: 179, Len: 61							
▶ EtherNet/IP (Industrial Protocol), Session: 0x10EEC402, Send Unit Data							
▶ Common Industrial Protocol							
▼ CIP Class Generic							
▼ Command Specific Data							
Data: 074d00f30a60054f100200							
0000	00 0c 29 56 20 17 00 1d	9c a1 28 4c 00 00 45 00	..)V(L..E.				
0010	00 65 55 a8 00 00 00 06	63 21 c0 a8 00 3b c0 a8	.eU.....c!...;..				
0020	00 3e af 12 d6 60 4d 64	18 72 c5 75 d3 41 50 18	.>....M d .r.u.AP.				
0030	07 d0 ef a9 00 00 70 00	25 00 d2 c4 ee 10 00 00P. %.....				
0040	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00				
0050	00 00 00 00 02 00 a1 00	04 00 01 00 fe 00 b1 00				
0060	11 00 01 00 cb 00 00 00	07 4d 00 f3 0a 60 05 4fM...'.0				
0070	10 02 00					

Figure 150. Restart Response over TCP (Functionality Test)

I. PCCC DOWNLOAD COMPLETED TEST CASES

This section shows the results of the PCCC Download Completed test cases.

(1) T72 Results

No.	Time	Source	Destination	Protocol	Length	Resp	Info
10	21:05:45.641635	192.168.0.59	192.168.0.62	CIP CM	124		Success: Connection Manager - Forward Open
11	21:05:45.655829	192.168.0.62	192.168.0.59	CIP	118		Class (0x67) - Service (0x4b)
12	21:05:45.661024	192.168.0.59	192.168.0.62	CIP	116		Success: Class (0x67) - Service (0x4b)
13	21:05:45.673622	192.168.0.62	192.168.0.59	CIP	118		Class (0x67) - Service (0x4b)
▶ Frame 12: 116 bytes on wire (928 bits), 116 bytes captured (928 bits) on interface 0							
▶ Ethernet II, Src: Rockwell_a1:28:4c (00:1d:9c:a1:28:4c), Dst: Vmware_56:20:17 (00:0c:29:56:20:17)							
▶ Internet Protocol Version 4, Src: 192.168.0.59, Dst: 192.168.0.62							
▶ Transmission Control Protocol, Src Port: 44818, Dst Port: 42022, Seq: 99, Ack: 179, Len: 62							
▶ EtherNet/IP (Industrial Protocol), Session: 0x529033E3, Send Unit Data							
▶ Common Industrial Protocol							
▼ CIP Class Generic							
▼ Command Specific Data							
Data: 074d00f30a60054ff002000b							
0000	00 0c 29 56 20 17 00 1d	9c a1 28 4c 00 00 45 00	..)V(L..E.				
0010	00 66 83 f5 00 00 00 06	34 d3 c0 a8 00 3b c0 a8	.f.....4.....				
0020	00 3e af 12 a4 26 4d 78	21 3d d5 d7 2c 01 50 18	.>....&Mx !=...;..P.				
0030	07 d0 1b 26 00 00 70 00	26 00 e3 33 90 52 00 00	...&..p. &..3.R..				
0040	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00				
0050	00 00 00 00 02 00 a1 00	04 00 01 00 fe 00 b1 00				
0060	12 00 01 00 cb 00 00 00	07 4d 00 f3 0a 60 05 4fM...'.0				
0070	f0 02 00 0b					

Figure 151. Download Completed Response over TCP (Functionality Test)

J. PCCC PROTECTED LOGICAL READ WITH THREE ADDRESS FIELDS TEST CASES

(1) T73 Results

For comparison, Figure 152 illustrates a Protected Logical Read with Three Address Fields request packet with File No. 0x03 and File Type 0x47 field inputs sent to a MicroLogix 1100 PLC. The SUT enters a fault state upon receiving the packet, i.e., no CIP response is observed. Figure 153 illustrates a similar request with identical File No. and File Type fields sent to the ControlLogix PLC. Figure 154 displays the ControlLogix PLC's response to the test packet. The ControlLogix does not fault. The response packet contains an EXT STS code of 0x06.

No.	Time	Source	Destination	Protocol	Length	Resp	Info
6	17:01:11.409602	192.168.0.62	192.168.0.59	TCP	60		47162 → 44818 [ACK] Seq=29 Ack=29 Win=29200 Len=0
7	17:01:11.418985	192.168.0.62	192.168.0.59	CIP CM	140		Connection Manager - Forward Open (Message Router)
8	17:01:11.425277	192.168.0.59	192.168.0.62	CIP CM	124		Success: Connection Manager - Forward Open
9	17:01:11.435849	192.168.0.62	192.168.0.59	CIP	123		Class (0x67) - Service (0x4b)
10	17:01:11.642151	192.168.0.62	192.168.0.59	TCP	123		[TCP Retransmission] 47162 → 44818 [PSH, ACK] Seq=115 Ack=99
11	17:01:11.850755	192.168.0.62	192.168.0.59	TCP	123		[TCP Retransmission] 47162 → 44818 [PSH, ACK] Seq=115 Ack=99
12	17:01:12.266704	192.168.0.62	192.168.0.59	TCP	123		[TCP Retransmission] 47162 → 44818 [PSH, ACK] Seq=115 Ack=99
▶ Frame 9: 123 bytes on wire (984 bits), 123 bytes captured (984 bits) on interface 0 ▶ Ethernet II, Src: Vmware_56:20:17 (00:0c:29:56:20:17), Dst: Rockwell_a1:28:4c (00:1d:9c:a1:28:4c) ▶ Internet Protocol Version 4, Src: 192.168.0.62, Dst: 192.168.0.59 ▶ Transmission Control Protocol, Src Port: 47162, Dst Port: 44818, Seq: 115, Ack: 99, Len: 69 ▶ Ethernet/IP (Industrial Protocol), Session: 0xb85ED9AD, Send Unit Data ▶ Common Industrial Protocol ▼ CIP Class Generic ▼ Command Specific Data Data: 074d00f30a60050f000200a2c80347bc00							
0000	00 1d 9c a1 28 4c 00 0c	29 56 20 17 00 00 45 00(L..)V...E.				
0010	00 6d d9 b4 40 00 40 06	df 0c c0 a8 00 3e c0 a8	.m..@..>...				
0020	00 3b b8 3a af 12 a0 82	80 bf 4d 59 21 4c 50 18	.;:.....MYILP.				
0030	72 10 a5 65 00 00 70 00	2d 00 ad d9 5e b8 00 00	r..e..p.~...^....				
0040	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00				
0050	00 00 14 00 02 00 a1 00	04 00 5e 53 d9 ad b1 00^S.....				
0060	19 00 01 00 4b 02 20 67	24 01 07 4d 00 f3 0a 60K. g S..M....				
0070	05 0f 00 02 00 a2 c8 03	47 bc 00G..				

Figure 152. MicroLogix Protected Logical Read with Three Address Fields Request over TCP (File No. 0x03 and File Type 0x47)

No.	Time	Source	Destination	Protocol	Length	Ethernet	Info
523	3.798362	10.1.100.4	10.1.40.1	CIP	116	Yes	Success: Class (0x67) - Service (0x4b)
524	3.805555	10.1.40.1	10.1.100.4	CIP	123	Yes	Class (0x67) - Service (0x4b)
525	3.805857	10.1.100.4	10.1.40.1	TCP	60	Yes	44818-39780 [ACK] Seq=10639 Ack=11916 Win=8123 Len
526	3.808605	10.1.100.4	10.1.40.1	CIP	116	Yes	Success: Class (0x67) - Service (0x4b)
527	3.815615	10.1.40.1	10.1.100.4	CIP	123	Yes	Class (0x67) - Service (0x4b)
▶ Frame 524: 123 bytes on wire (984 bits), 123 bytes captured (984 bits) on interface 0 ▶ Ethernet II, Src: IntelCor_18:fc:72 (90:e2:ba:18:fc:72), Dst: Rockwell_cd:46:e3 (00:1d:9c:cd:46:e3) ▶ Internet Protocol Version 4, Src: 10.1.40.1, Dst: 10.1.100.4 ▶ Transmission Control Protocol, Src Port: 39780, Dst Port: 44818, Seq: 11847, Ack: 10639, Len: 69 ▶ EtherNet/IP (Industrial Protocol), Session: 0x00030001, Send Unit Data ▶ Common Industrial Protocol ▼ CIP Class Generic ▼ Command Specific Data Data: 074d00f30a60050f00ac00a20103470000							
0000	00 1d 9c cd 46 e3 90 e2	ba 18 fc 72 08 00 45 00F... ..r..E.				
0010	00 6d 18 c7 40 00 40 06	81 bd 0a 01 28 01 0a 01	.m..@.@.(...				
0020	64 04 9b 64 af 12 b6 78	f3 2b db 15 33 14 50 18	d..d...x .+.3.P.				
0030	00 e5 a0 66 00 00 70 00	2d 00 01 00 03 00 00 00	...f..p. ~.....				
0040	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00				
0050	00 00 14 00 02 00 a1 00	04 00 02 40 92 ff b1 00@.....				
0060	19 00 ab 00 4b 02 20 67	24 01 07 4d 00 f3 0a 60K. g \$.M....`				
0070	05 0f 00 ac 00 a2 01 03	47 00 00 G..				

Figure 153. ControlLogix Protected Logical Read with Three Address Fields Request over TCP (File No. 0x03 and File Type 0x47)

No.	Time	Source	Destination	Protocol	Length	Ethernet	Info
524	3.805555	10.1.40.1	10.1.100.4	CIP	123	Yes	Class (0x67) - Service (0x4b)
525	3.805857	10.1.100.4	10.1.40.1	TCP	60	Yes	44818-39780 [ACK] Seq=10639 Ack=11916 Win=8123 Len
526	3.808605	10.1.100.4	10.1.40.1	CIP	116	Yes	Success: Class (0x67) - Service (0x4b)
527	3.815615	10.1.100.4	10.1.40.1	CIP	123	Yes	Class (0x67) - Service (0x4b)
528	3.815852	10.1.100.4	10.1.40.1	TCP	60	Yes	44818-39780 [ACK] Seq=10701 Ack=11985 Win=8123 Len
▶ Frame 526: 116 bytes on wire (928 bits), 116 bytes captured (928 bits) on interface 0 ▶ Ethernet II, Src: Rockwell_cd:46:e3 (00:1d:9c:cd:46:e3), Dst: IntelCor_18:fc:72 (90:e2:ba:18:fc:72) ▶ Internet Protocol Version 4, Src: 10.1.100.4, Dst: 10.1.40.1 ▶ Transmission Control Protocol, Src Port: 44818, Dst Port: 39780, Seq: 10639, Ack: 11916, Len: 62 ▶ EtherNet/IP (Industrial Protocol), Session: 0x00030001, Send Unit Data ▶ Common Industrial Protocol ▼ CIP Class Generic ▼ Command Specific Data Data: 074d00f30a60054ff0ac0006							
0000	90 e2 ba 18 fc 72 00 1d	9c cd 46 e3 08 00 45 00r... ..F...E.				
0010	00 66 a2 05 40 00 40 06	f8 85 0a 01 64 04 0a 01	.f..@.@.d...				
0020	28 01 af 12 9b 64 db 15	33 14 b6 78 f3 70 50 18	(....d.. 3..x.pP.				
0030	20 00 6a d6 00 00 70 00	26 00 01 00 03 00 00 00	.j...p. &.....				
0040	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00				
0050	00 00 00 00 02 00 a1 00	04 00 01 00 fe 80 b1 00				
0060	12 00 ab 00 cb 00 00 00	07 4d 00 f3 0a 60 05 4fM....`.0				
0070	f0 ac 00 06					

Figure 154. ControlLogix Protected Logical Read with Three Address Fields Response over TCP (File No. 0x03 and File Type 0x47)

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

- [1] J. Slay and M. Michael, "Lessons learned from the Maroochy Water breach," in *Critical Infrastructure Protection*, Boston, MA: Springer, 2008, pp. 73–82.
- [2] J. R. Lindsay, "Stuxnet and the limits of cyber warfare," *Security Studies*, vol. 22, pp. 365–404, 2013.
- [3] K. Zetter, "Inside the cunning, unprecedented hack of Ukrain's power grid," March 3, 2016. [Online]. Available: <https://www.wired.com/2016/03/inside-cunning-unprecedented-hack-ukraines-power-grid/>
- [4] S. Young, "Dallas siren hack came via radio frequency, not computer, city says," April 10, 2017. [Online]. Available: www.dallasobserver.com/news/dallas-siren-hack-done-by-radio-not-computer-9358087
- [5] M. Swearingen, S. Brunasso, J. Weiss and D. Huber, "What you need to know (and don't) about the AURORA vulnerability," *Power Magazine*, September 1, 2013. [Online]. Available: <http://www.powermag.com/what-you-need-to-know-and-dont-about-the-aurora-vulnerability/?pagenum=2>
- [6] J. Mulder, M. Schwartz, M. Berg, J. R. Van Houten, J. M. Urrea, M. A. King, A. A. Clements and J. Jacob, "WeaselBoard: Zero-day exploit detection for programmable logic controllers," Albuquerque, NM, 2013.
- [7] D. G. Peterson, "Digital bond," January 19, 2012. [Online]. Available: <http://www.digitalbond.com/blog/2012/01/19/project-basecamp-at-s4/>
- [8] B. P. Miller, L. Fredriksen and B. So, "An empirical study of the reliability of UNIX utilities," *Communications of the ACM* 33, vol. 12, December 1990.
- [9] F. Tacliad, "ENIP fuzz: A scapy-based EtherNet/IP fuzzer for security testing," M.S. thesis, Dept. of Computer Science., NPS, Monterey, CA, USA, 2016.
- [10] M. Henry, M. Iacovelli and J. Thatcher, "DDG 1000 engineering control system (ECS)." [Online]. Available: https://seagrant.mit.edu/ESRDC_library/Henry-DDG-1000.pdf
- [11] ICS-CERT, "Ongoing sophisticated malware campaign compromising ICS (update E)," 2014. [Online]. Available: <https://ics-cert.us-cert.gov/alerts/ICS-ALERT-14-281-01B>
- [12] B. Freeman, "A new defense for Navy ships: Protection from cyber attacks," September 17, 2015. [Online]. Available: <https://www.onr.navy.mil/en/Media-Center/Press-Releases/2015/RHIMES-Cyber-Attack-Protection.aspx>

- [13] R. Grandgenett, R. Gandhi and W. Mahoney, "Exploitation of Allen Bradley's implementation of EtherNet/IP for denial of service against industrial control systems," in *9th International Conference on Cyber Warfare and Security*, Purdue University, 2014.
- [14] R. Grandgenett, W. Mahoney, and R. Gandhi, "Authentication bypass and remote escalated I/O command attacks," in *Proceedings of the 10th Annual Cyber and Information Security Research Conference*, 2015.
- [15] ODVA & ControlNet International, Ltd., *The CIP Networks Library. Vol. 1. Common Industrial Protocol (CIP)*, 3.22 ed., Ann Arbor, MI: Open DeviceNet Vendor Association, Inc., 2017.
- [16] ODVA & ControlNet International, Ltd., *The CIP Networks Library Volume 2: EtherNet/IP Adaptation of CIP*, 1.23 ed., Ann Arbor, MI: Open DeviceNet Vendor Association, Inc., 2017.
- [17] Allen Bradley, *DF1 Protocol and Command Set Reference Manual*, Milwaukee, WI, USA, October 1996.
- [18] SECDEV, "Scapy," accessed April 3, 2017. [Online]. Available: <http://www.secdev.org/projects/scapy/>
- [19] P. Brooks, "EtherNet/IP: Industrial protocol white paper," October 2001. [Online]. Available: http://literature.rockwellautomation.com/idc/groups/literature/documents/wp/enet-wp001_-en-p.pdf
- [20] V Schiffer on behalf of Rockwell Automation, "The Common Industrial Protocol (CIP) and the Family of CIP Networks," ODVA, Inc., Ann Arbor, MI, USA, 2016.
- [21] Rockwell Automation, "Communicating with RA Products Using EtherNet/IP Explicit Messaging (Rev 1.2)," Rockwell Automation, 2001. [Online]. Available: https://www.rockwellautomation.com/resources/downloads/rockwellautomation/pdf/sales-partners/technology-licensing/eipexp1_2.pdf
- [22] ODVA, "The Common Industrial Protocol (CIP)," 2017. [Online]. Available: <https://www.odva.org/Technology-Standards/Common-Industrial-Protocol-CIP/Overview>
- [23] ODVA, "Common Industrial Protocol (CIP)," 2006. [Online]. Available: https://scadahacker.com/library/Documents/ICS_Protocols/ODVA%20-%20CIP.pdf

- [24] Rockwell Automation, “Delivery of CIP over RA Serial DF1 Links (Rev. 1.1),” Rockwell Automation, 2006. [Online]. Available: https://www.rockwellautomation.com/resources/downloads/rockwellautomation/pdf/sales-partners/technology-licensing/CIPandPCCC_v1_1.pdf
- [25] N. Champey, “Principles of EtherNet/IP Communication,” accessed April 24, 2017. Rueil-Malmaison, FR. [Online]. Available: https://scadahacker.com/library/Documents/ICS_Protocols/Schneider%20-%20Principles%20of%20EtherNetIP%20Communication.pdf
- [26] Allen-Bradley / Rockwell Automation, “Enhanced DeviceNet communications module,” May 2000. Milwaukee, WI, USA. [Online]. Available: http://literature.rockwellautomation.com/idc/groups/literature/documents/um/1203-um014_-en-p.pdf
- [27] R. Shapiro, S. Bratus, E. Rogers, and S. Smith, “Do-it-yourself SCADA vulnerability testing with LZFUZZ,” accessed April 15, 2017. [Online]. Available: <http://www.cs.dartmouth.edu/~sws/pubs/sbs11.pdf>
- [28] D. Aitel, “The advantages of block-based protocol analysis for security testing,” February 4, 2002. [Online]. Available: http://www.immunitysec.com/downloads/advantages_of_block_based_analysis.pdf
- [29] Beyond Security, “Dynamic testing (fuzzing) on the Ethernet IP Protocol by beSTORM Ethernet IP with beSTORM,” accessed April 25, 2017. [Online]. Available: http://www.beyondsecurity.com/dynamic_fuzzing_testing_ethernet_IP_protocol
- [30] A. Portnoy, P. Amini and R. Sears. “Sulley,” accessed April 20, 2017. [Online]. Available: <https://github.com/OpenRCE/sulley>
- [31] S. Bansal and N. Bansal, “Scapy—A Python tool for security testing,” *Journal of Computer Science & Systems Biology*, March 31, 2015.
- [32] P. Joshi, P. Patel, and R. Parikh, “A fuzz testing framework for Wi-Fi devices,” *International Journal of Research in Engineering & Advanced Technology*, vol. 3, no. 5, October-November 2015.
- [33] H. Rafiee, C. Mueller, L. Niemeier, J. Streek, C. Sterz, and C. Meinel, “A flexible framework for detecting IPv6 vulnerabilities,” in *The 6th International Conference on Security of Information Networks*, Aksaray, Turkey, 2013.
- [34] A. Lahmadi, C. Bernardini, and O. Festor, “A testing framework for discovering vulnerabilities in 6LoWPAN networks,” in *8th International Conference on Distributed Computing in Sensor Systems (DCOSS2012)*, Hangzhou, China., 2012.

- [35] P. Tsankov, T. Dashti, and D. Basin, "SECFUZZ: Fuzz-testing security protocols," in *7th International Workshop on Automation of Software Test*, Zurich, Switzerland, 2012.
- [36] A. G. Voyiatzis, K. Katsigiannis, and S. Koubias, *A Modbus/TCP Fuzzer for Testing Internetworked Industrial Systems*, IEEE, 2015.
- [37] MODICON, Inc., Industrial Automation Systems, *Modicon Modbus Protocol Reference Guide*, North Andover, MA: MODICON, Inc., Industrial Automation Systems, 1996.
- [38] R. Sprabery, T. Morris, and S. P. V. Madani, "Protocol mutation intrusion detection for synchrophasor communications," in *Eighth Annual Cyber Security and Information Intelligence Research Workshop*, Oak Ridge, TN, 2012.
- [39] Rockwell Automation, "MicroLogix Programmable Controllers Selection Guide," Milwaukee, WI, USA, 2015. Accessed January 24, 2017. [Online]. Available: http://literature.rockwellautomation.com/idc/groups/literature/documents/sg/1761-sg001_-en-p.pdf
- [40] Allen-Bradley Rockwell Automation, "ControlLogix 5570 Controllers," accessed January 23, 2017. [Online]. Available: <http://ab.rockwellautomation.com/Programmable-Controllers/ControlLogix/5570-Controllers#overview>
- [41] ICS-CERT, "Advisory (ICSA-17-138-03) Rockwell Automation MicroLogix 1100 Controllers," July 18, 2017. [Online]. Available: <https://ics-cert.us-cert.gov/advisories/ICSA-17-138-03>

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
Ft. Belvoir, Virginia
2. Dudley Knox Library
Naval Postgraduate School
Monterey, California